Procesamiento de datos

Presentación con una selección de las slides de las dos primeras presentaciones de Chip Huyen que se pueden consultar en [*stanford-cs329s.github.io/syllabus.html*](stanford-cs329s.github.io/syllabus.html)

1. Mind vs. Data

WHO WOULD WIN?

Intelligent model architectures that took researchers their entire PhDs to design

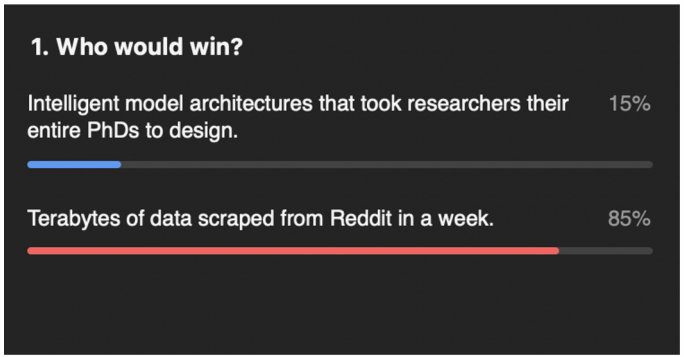
Terabytes of data scraped from Reddit in a week

Poll: who would win?

1. Intelligent design
2. TB of Reddit data

WHO WOULD WIN?

Intelligent model architectures that took researchers their entire PhDs to design



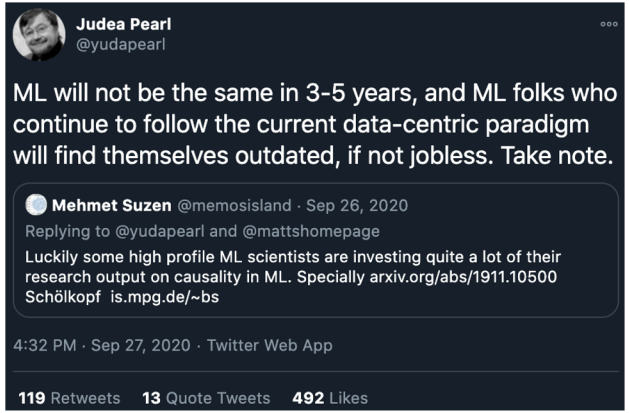
Terabytes of data scraped from Reddit in a week

From last year

Mind

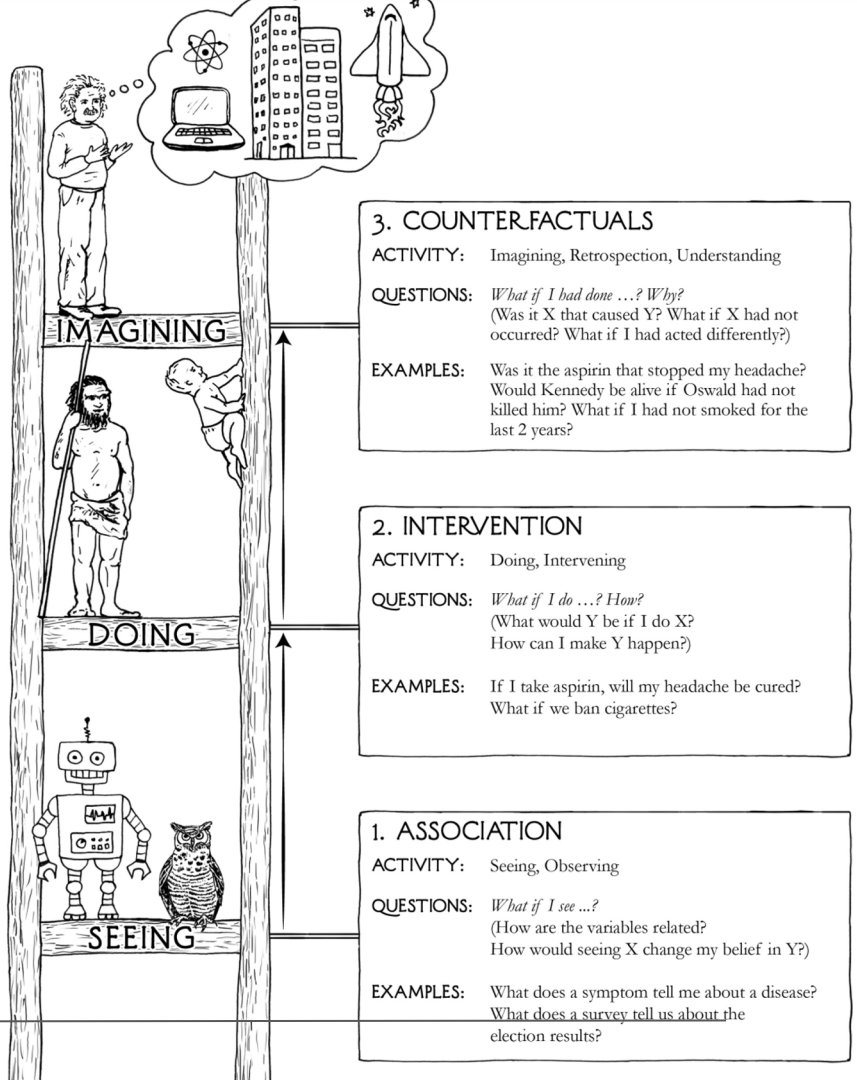
“Data is profoundly dumb.”

Judea Pearl, *Mind over data - The Book of Why*



Limitations of data

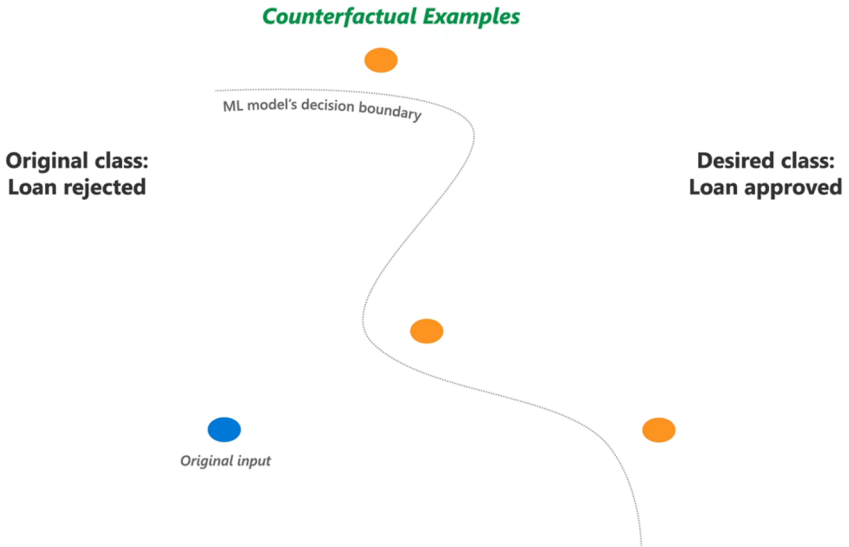
- Observed X, output Y
- Can learn P(X, Y)
- What would happen if didn't do X?



The Book of Why (Pearl and MacKenzie, 2018)

Counterfactuals: example

- ML predicts loan app rejection
- What do I need to change to get my loan approved?

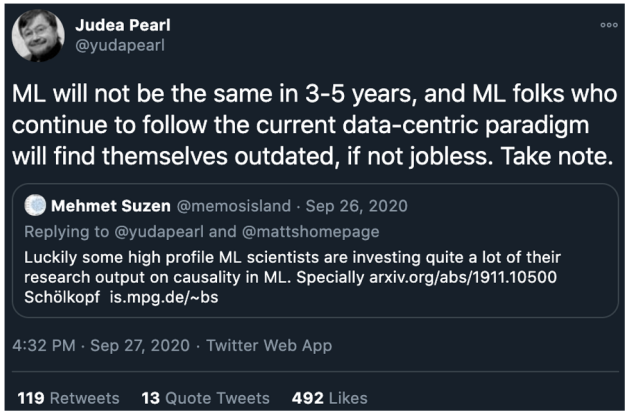


Open-source library provides explanation for machine learning through diverse counterfactuals (Amit Sharma, Microsoft 2020)

Mind

“Data is profoundly dumb.”

Judea Pearl, *Mind over data - The Book of Why*



Data

“General methods that leverage computation are ultimately the most effective, and by a large margin … Human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation.”

Richard Sutton, *Bitter Lesson*

“We don’t have better algorithms. We just have more data.”

Peter Norvig, *The Unreasonable Effectiveness of Data*

“Imposing structure requires us to make certain assumptions, which are invariably wrong for at least some portion of the data.”

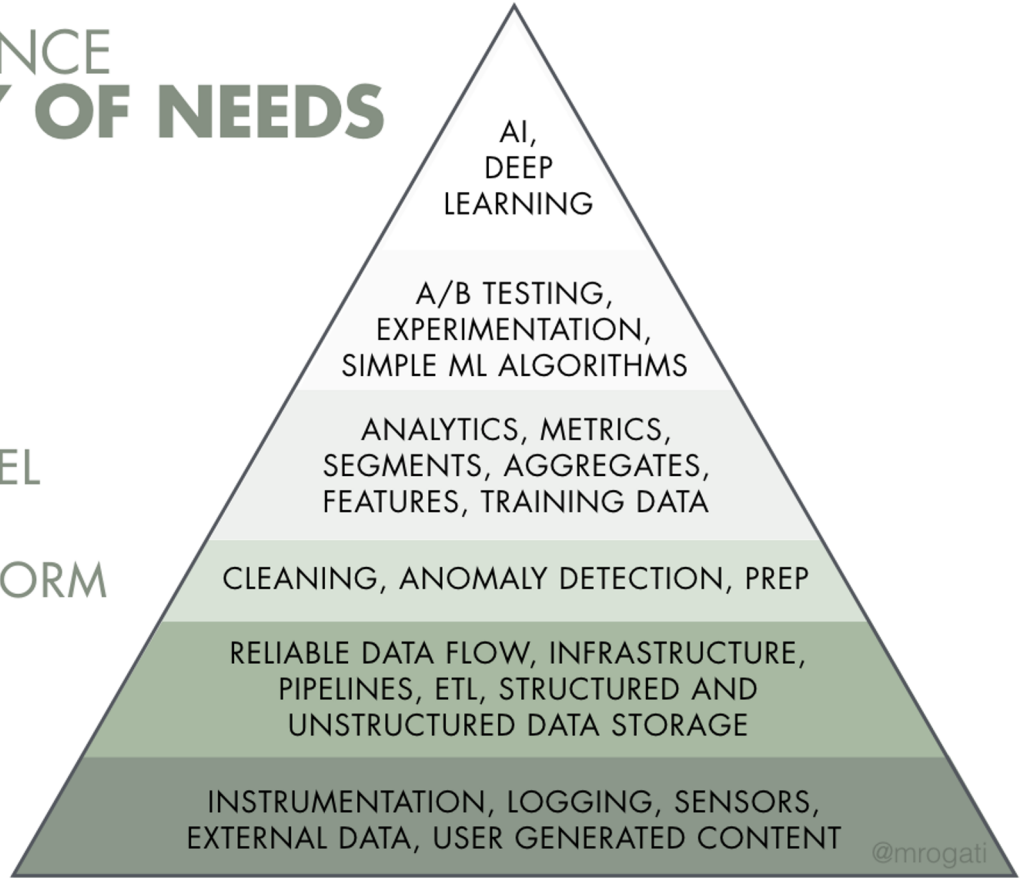
Yann LeCun, *Deep Learning and Innate Priors*

Data is necessary.
The debate is whether *finite** data is sufficient.

* If we had infinite data, we can solve arbitrarily complex problems by just looking up the answers.

Massive data $\neq$ infinite data

THE DATA SCIENCE HIERARCHY OF NEEDS



LEARN/OPTIMIZE

AGGREGATE/LABEL

EXPLORE/TRANSFORM

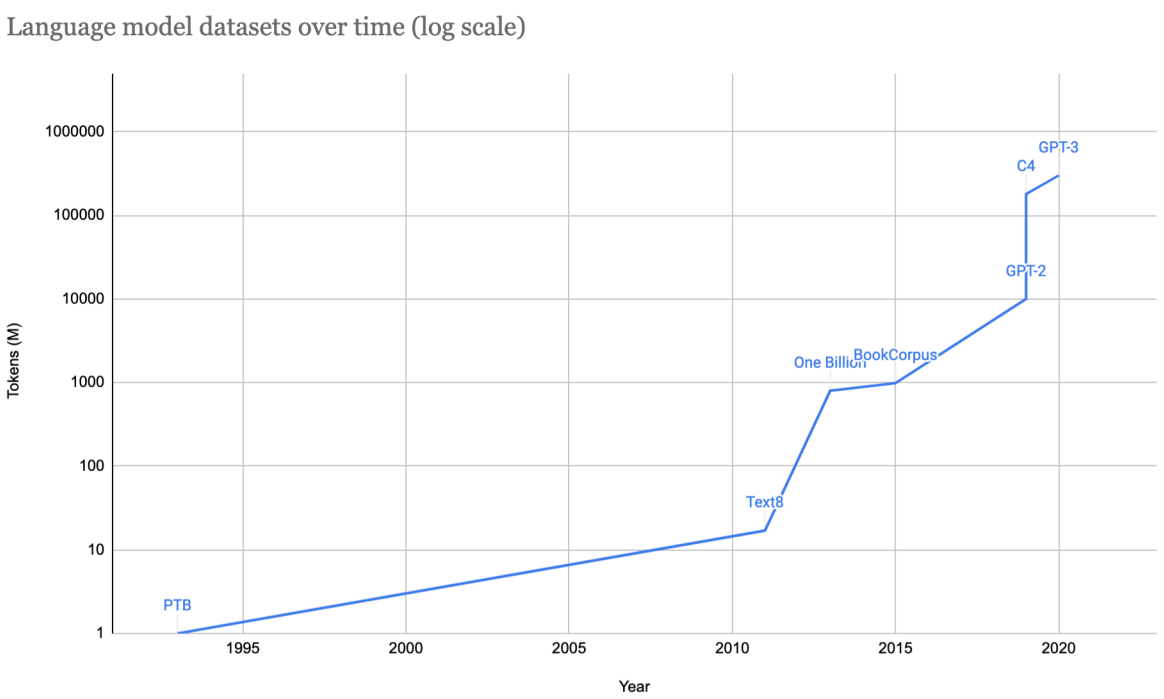
MOVE/STORE

COLLECT

The AI Hierarchy of Needs (Monica Rogati, 2017)

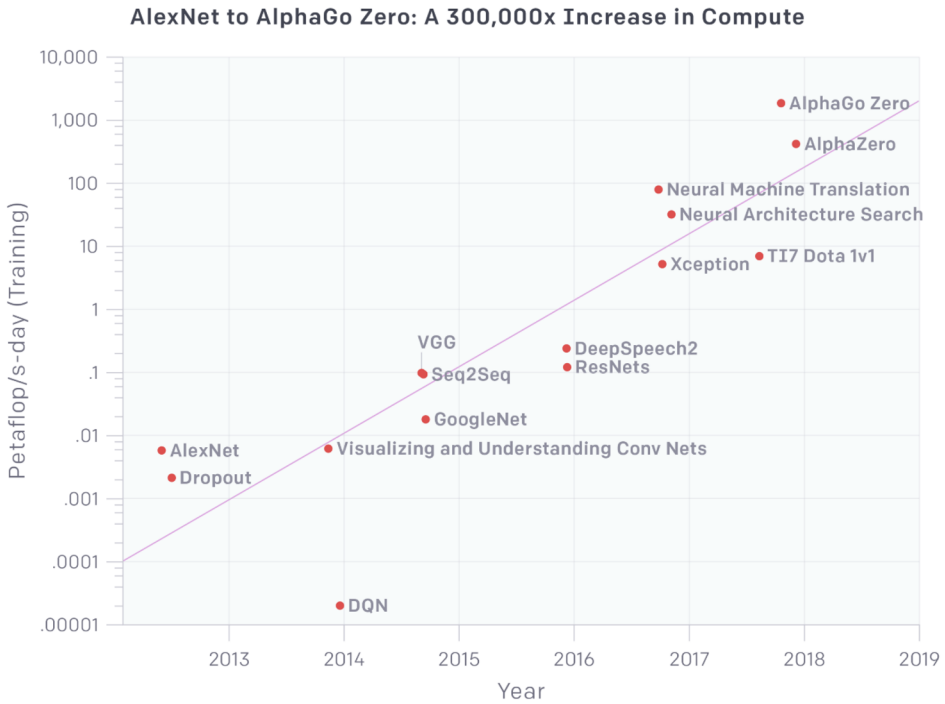
Datasets for language models

Year	Tokens (M)	Dataset
1993	1	PTB
2011	17	Text8
2013	800	One Billion
2015	985	BookCorpus
2019	10,000	GPT-2
2019	180,000	C4
2020	300,000	GPT-3



More data (generally) needs more compute

"amount of compute used in the largest AI training runs has doubled every 3.5 months"



AI and Compute (OpenAI 2018)

⚠️ Data: full of potential for biases ⚠️

- sampling/selection biases
- under/over-representation of subgroups
- human biases embedded in historical data
- labeling biases
- ...

Algorithmic biases not covered (yet)!

2. Labeling

Labeling

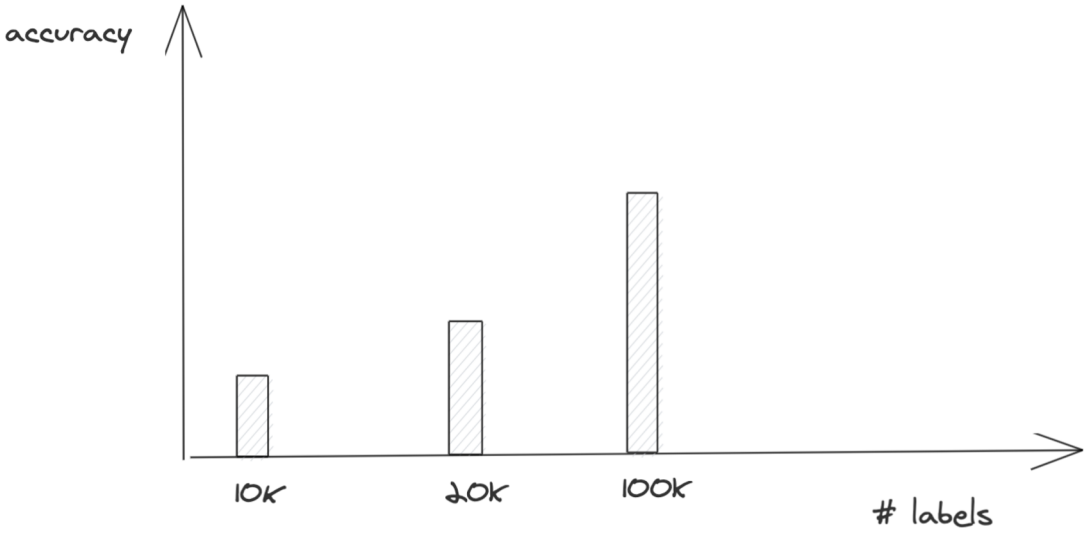
When I told our recruiters that I wanted an in-house labeling team, they asked how long I'd need this team for. I told them: "How long do we need an engineering team for?"

Andrej Karpathy, Director of AI @ Tesla
[CS 329S guest lecture, 2021]

Labeling

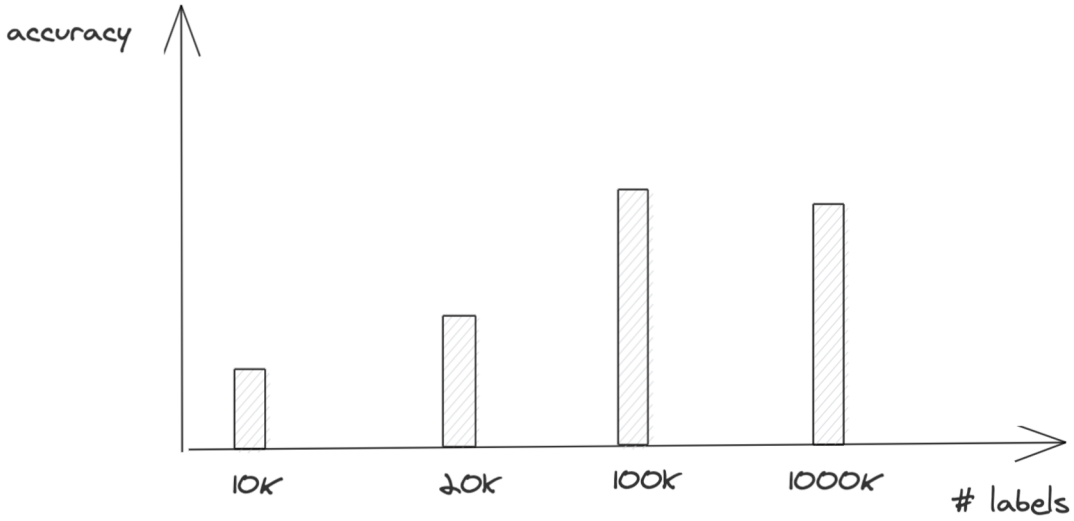
1. Hand-labeling
2. Programmatic labeling
3. Weak supervision, semi supervision, active learning, transfer learning

⚠️ More data isn't always better ⚠️



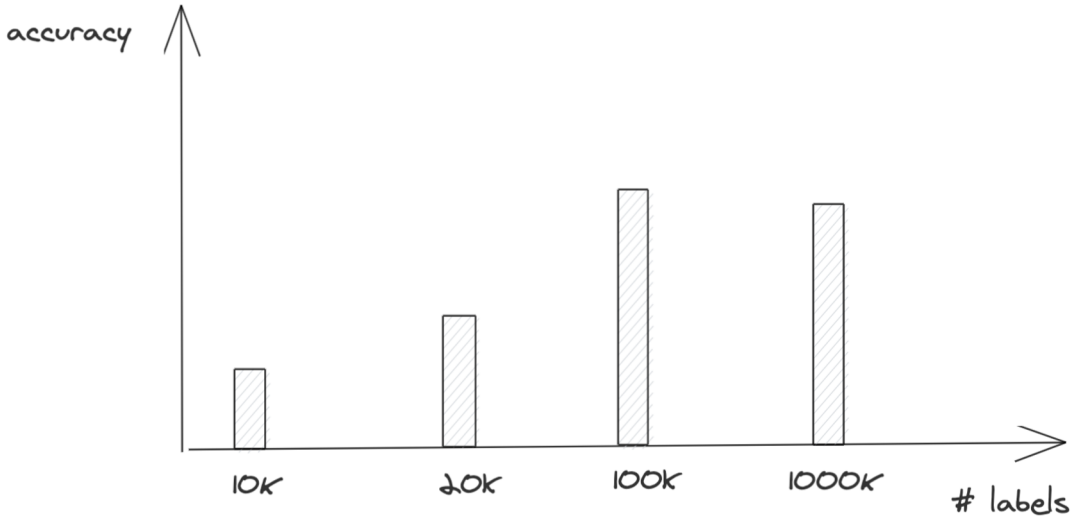
🧠 Idea 🧠: crowdsource data to get 1 million labels!

⚠️ More data isn't always better ⚠️



Why is the model getting worse?

⚠️ Label sources with varying accuracy ⚠️



- 100K labels: internally labeled, high accuracy
- 1M labels: crowdsourced, noisy

Label multiplicity: example

Task: label all entities in the following sentence:

Darth Sidious, known simply as the Emperor, was a Dark Lord of the Sith
who reigned over the galaxy as Galactic Emperor of the First Galactic Empire.

Label multiplicity: example

Zoom poll: which annotator is correct?

Task: label all entities in the following sentence:

Darth Sidious, known simply as the Emperor, was a Dark Lord of the Sith
who reigned over the galaxy as Galactic Emperor of the First Galactic Empire.

Annotator	# entities	Annotation
1	3	[**Darth Sidious**], known simply as the Emperor, was a [**Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**]
2	6	[**Darth Sidious**], known simply as the [**Emperor**], was a [**Dark Lord**] of the [**Sith**] who reigned over the galaxy as [**Galactic Emperor**] of the [**First Galactic Empire**].
3	4	[**Darth Sidious**], known simply as the [**Emperor**], was a [**Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**].

Label multiplicity

More expertise required (more difficult to label), more room for disagreement!

If experts can’t agree on a label, time to rethink human-level performance

Label multiplicity: solution

- Clear problem definition
  - Pick the entity that comprises the longest substring

Annotator	# entities	Annotation
1	3	[**Darth Sidious**], known simply as the Emperor, was a [**Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**]
2	6	[**Darth Sidious**], known simply as the [**Emperor**], was a [**Dark Lord**] of the [**Sith**] who reigned over the galaxy as [**Galactic Emperor**] of the [**First Galactic Empire**].
3	4	[**Darth Sidious**], known simply as the [**Emperor**], was a [**Dark Lord of the Sith**] who reigned over the galaxy as [**Galactic Emperor of the First Galactic Empire**].

Label multiplicity: solution

- Clear problem definition
- Annotation training
- Data lineage: track where data/labels come from

Label multiplicity: solution

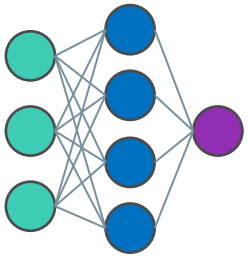
- Clear problem definition
- Annotation training
- Data lineage: track where data/labels come from
- Learning methods with noisy labels
  - Learning with Noisy Labels (Natarajan et al., 2013)
  - Loss factorization, weakly supervised learning and label noise robustness (Patrini et al., 2016)
  - Cost-Sensitive Learning with Noisy Labels (Natarajan et al., 2018)
  - Confident Learning: Estimating Uncertainty in Dataset Labels (Northcutt et al., 2019)

Programmatic labeling

Training data is the bottleneck

Data

Algorithms



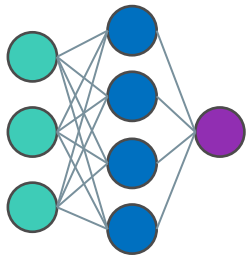
ML Model

Training data is the bottleneck

Data

Key differentiator

Algorithms



```
from transformers \
  import BertModel as model
```

Increasingly commoditized

ML Model

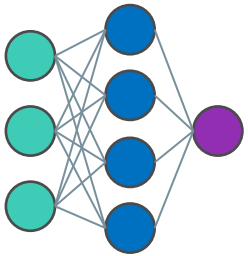
"We don't have better algorithms. We just have more data."

Peter Norvig, [The Unreasonable Effectiveness of Data]

Training data is the bottleneck

Data

Algorithms



ML Model

- 8 Person-months
- 8-9 pt. differences

- 1-2 days
- <1 pt. differences

Stanford MEDICINE

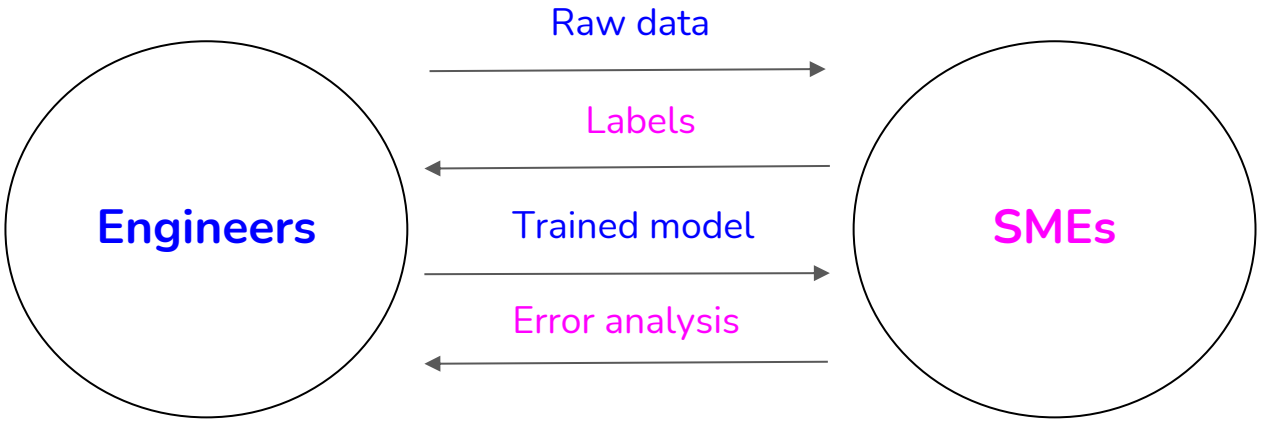
How to get training data in days?

Cross-Modal Data Programming Enables Rapid Medical Machine Learning (Dunnmon et al., 2019)

Hand labeling data is …

- **Expensive:** Esp. when **subject matter expertise** required
- **Non-private:** Need to ship data to human annotators
- **Slow:** Time required scales linearly with # labels needed
- **Non-adaptive:** Every change requires re-labeling the dataset

Cross-functional communication



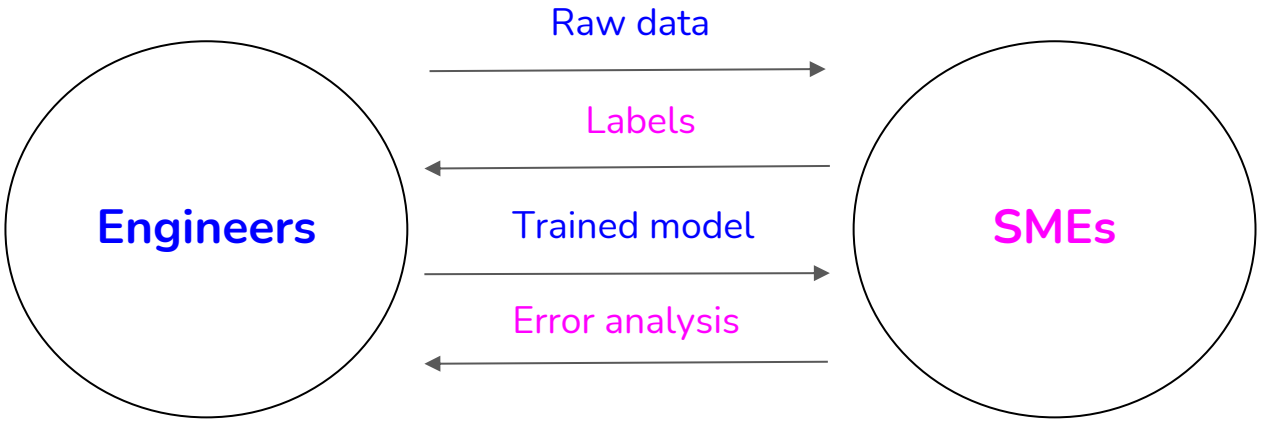

```
def function:
   if X:
           do Y
```

```
If the nurse's note mentions
serious conditions like pneumonia,
the patient's case should be given
priority consideration.
```

Code: version control, reuse, share

How to version, share, reuse **expertise**?

SME as labeling functions




```
def function:
   if X:
            do Y
```

```
def labeling_function(note):
   if “pneumonia” in note:
            return “EMERGENT”
```

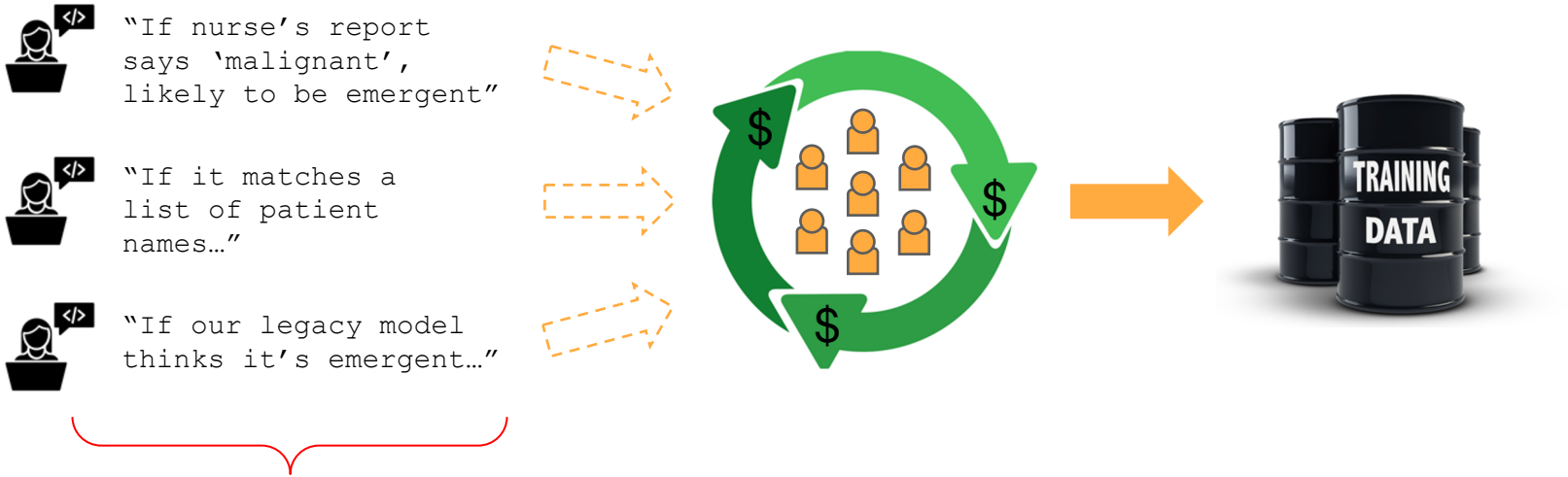
Labeling functions (LFs): Encode SME heuristics as function and use them to label training data *programmatically*

snorkel

LFs: can express many different types of heuristics

Pattern Matching	If a phrase like “send money” is in email
Boolean Search	If unknown_sender AND foreign_source
DB Lookup	If sender is in our Blacklist.db
Heuristics	If SpellChecker finds 3+ spelling errors
Legacy System	If LegacySystem votes spam
Third Party Model	If BERT labels an entity “diet”
Crowd Labels	If Worker #23 votes spam

LFs: can express many different types of heuristics



Labeling functions: Simple, flexible, interpretable, adaptable, fast

LFs: powerful but noisy

```
def LF_contains_money(x):
    if "money" in x.body.text:
        return "SPAM"
```

```
def LF_from_grandma(x):
    if x.sender.name is "Grandma":
        return "HAM"
```

```
def LF_contains_money(x):
    if "free money" in x.body.text:
        return "SPAM"
```

From: **Grandma**

"Dear handsome grandson,
Since you can't be home for Thanksgiving dinner this year, I'm sending you some **money** so you could enjoy a nice meal …"

??

"You have been pre-approved for free **cash** …"

??

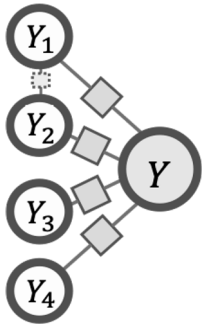
- **Noisy**: Unknown, inaccurate
- **Overlapping**: LFs may be correlated
- **Conflicting**: different LFs give different labels
- **Narrow**: Don't generalize well

LF labels are combined to generate ground truths

```
def LF_contains_money(x):
    if "money" in x.body.text:
        return "SPAM"
```

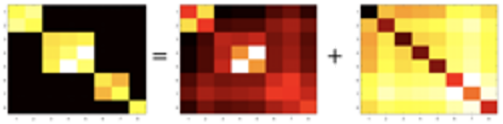
```
def LF_from_grandma(x):
    if x.sender.name is "Grandma":
        return "HAM"
```

```
def LF_contains_money(x):
    if "free money" in x.body.text:
        return "SPAM"
```



[Intuition]

Look at agreements & disagreements



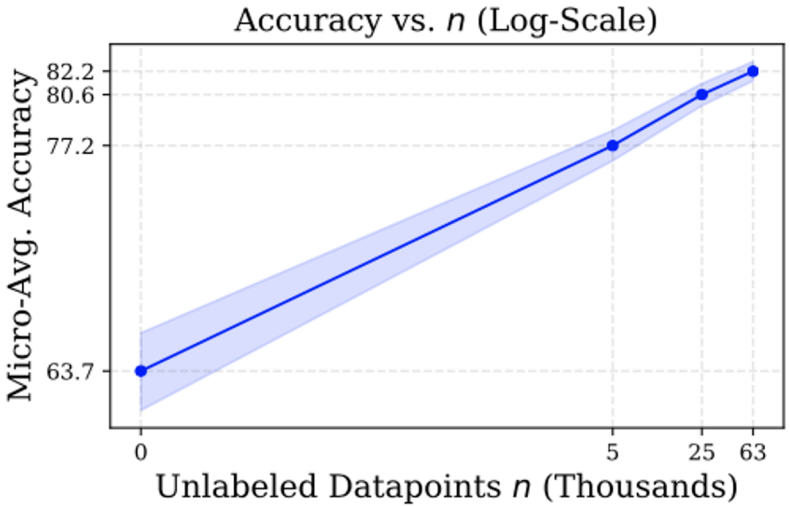
$$(\Sigma^{-1})_O = \Sigma_O^{-1} + zz^T$$

Provably consistent matrix completion-style algorithm over inverse covariance

[Ratner et. al. NeurIPS'16;
Bach et. al. ICML'17;
Ratner et. al. AAAI'19;
Varma et. al. ICML'19l;
Sala et. al. NeurIPS'19;
Fu et. al. ICML'20]

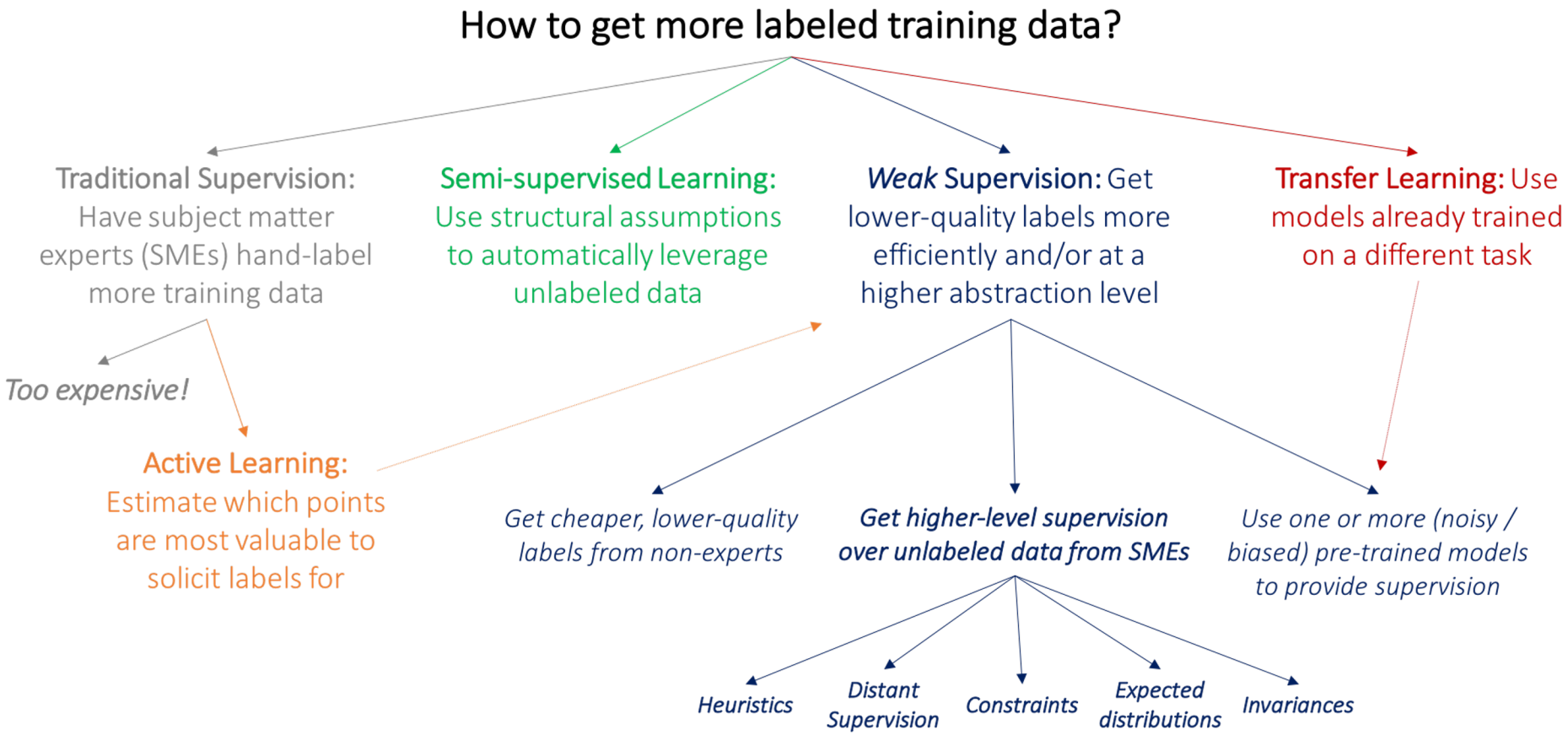
Hand labeling	Programmatic labeling
Expensive: esp. when subject matter expertise required	**Cost saving**: Expertise can be versioned, shared, reused across organization
Non-private: Need to ship data to human annotators	**Privacy**: Create LFs using a cleared data subsample then apply LFs to other data without looking at individual samples.
Slow: Time required scales linearly with # labels needed	**Fast**: Easily scale 1K -> 1M samples
Non-adaptive: Every change requires re-labeling the dataset	**Adaptive**: When changes happen, just reapply LFs!

Programmatic labeling: Scale with unlabeled data



Ratner et. al. NeurIPS'16; Ratner et. al. VLDB'18; Ratner et. al. AAAI'19

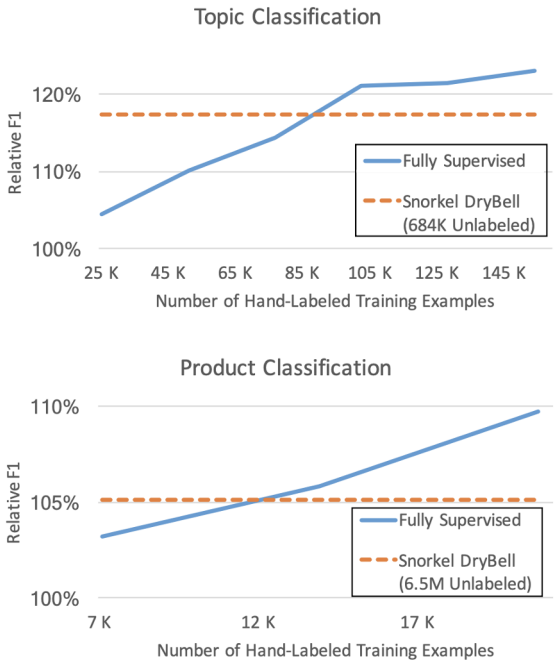
Weak supervision, semi-supervision,
active learning,
transfer learning



Weak Supervision: A New Programming Paradigm for Machine Learning (Ratner et al., 2019)

Weak supervision

- Leverage noisy, imprecise sources to create labels
  - e.g. if “money” is in an email it’s probably spam



Harnessing Organizational Knowledge for Machine Learning (Ratner et al., Google AI Blog 2019)

Semi-supervision

- Use structural assumptions to leverage a large amount of unlabeled data together with a small amount of labeled data
  - Hashtags in the same profile/tweet are probably of similar topics

Semi-supervision

- Use structural assumptions to leverage a large amount of unlabeled data together with a small amount of labeled data
- Might require complex algorithms like clustering to discover similarity

Semi-supervision: self-training

1. Train model on a small set of labeled data
2. Use this model to generate predictions for unlabeled data
3. Use predictions with high raw probabilities as labels
4. Repeat step 1 with new labeled data

Semi-supervision: perturbation-based methods

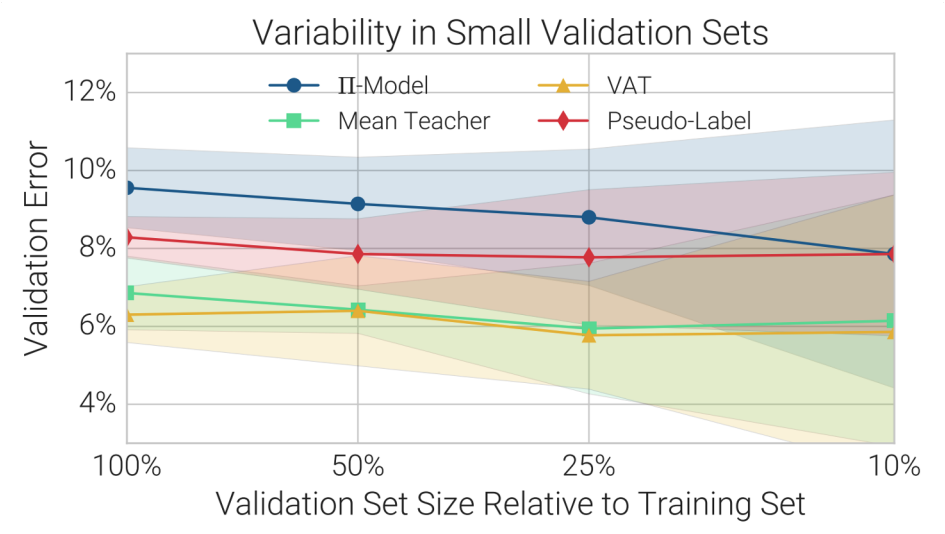
Assumption: small perturbation wouldn't change a sample's label

- Add white noises to images
- Add small values to word embeddings

Also a data augmentation method!

Semi-supervision challenge: valid set's size

- Big valid set: less data for training
- Small valid set: not enough signal to choose the best model



Realistic Evaluation of Deep Semi-Supervised Learning Algorithms (Oliver et al., 2018)

Transfer learning

Apply model trained for one task to another task

1. Fine-tuning
2. Prompt-based

Language Models are Few-Shot Learners (OpenAI 2020)

Transfer learning: fine-tuning

- fine-tuning only some layers
- fine-tuning the entire model

Language Models are Few-Shot Learners (OpenAI 2020)

Transfer learning: Prompt-based

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée        ←
plush girafe => girafe peluche      ←
cheese =>                           ← prompt
```


Language Models are Few-Shot Learners (OpenAI 2020)

Demo

Use GPT-3 to generate React apps
https://twitter.com/sharifshameem/status/1284095222939451393

Active learning

- **Goal**: Increase the efficiency of labels
- Label samples that are estimated to be most valuable to the model according to some metrics

Active Learning Literature Survey (Burr Settles, 2010)

Active learning metrics

- Uncertainty measurement
  - e.g. label samples with lowest raw probability for the predicted class

Active Learning Literature Survey (Burr Settles, 2010)

Active learning metrics

- Uncertainty measurement
- Candidate models' disagreement
  - Have several candidate models (e.g. models with different hypeparams)
  - Each model makes its own prediction
  - Label samples with most disagreement

Active Learning Literature Survey (Burr Settles, 2010)

Active learning

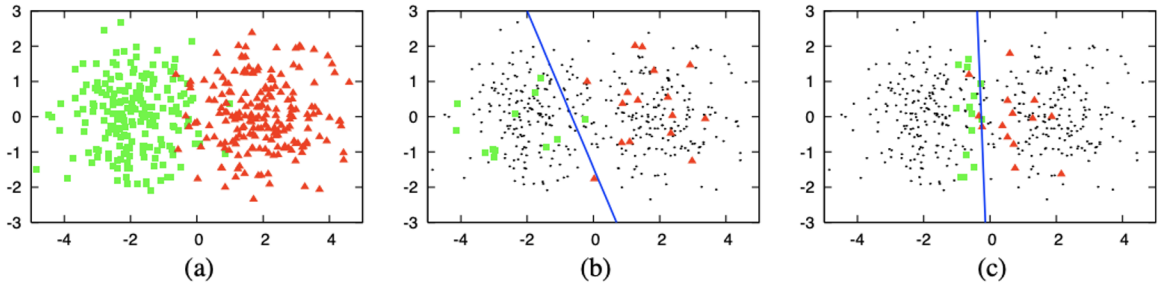


Figure 2: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier (70% accuracy). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (90%).

Active Learning Literature Survey (Burr Settles, 2010)

Method	How	Ground truths required?
Weak supervision	Leverages (often noisy) heuristics to generate labels	No, but a small number of labels is useful to guide the development of heuristics
Semi-supervision	Leverages structural assumptions to generate labels	Yes. A small number of initial labels as seeds to generate more labels
Transfer learning	Leverages models pretrained on another task for your new task	No for zero-shot learning Yes for fine-tuning, though # GTs required is often much less than # GTs required if training from scratch.
Active learning	Labels data samples that are most useful to your model	Yes

4. Sampling

Types of sampling

- Non-probability sampling
  - Convenience sampling: selection based on availability
    - Soliciting response
    - Choosing existing datasets
    - Looking at available reviews on Amazon
  - Snowball sampling: future samples are selected based on existing samples
    - E.g. to scape legit Twitter accounts, start with seed accounts then scrape their following
  - Judgment sampling: experts decide what to include
  - Quota sampling: quotas for certain slices of data (no randomization)
  - ....

Data used in ML is mostly driven by convenience

- Language models: BookCorpus, CommonCrawl, Wikipedia, Reddit links
- Sentiment analysis: IMDB, Amazon
  - Only users who have access to the Internet and are willing to put reviews online
- Self-driving cars: most data is from the Bay Area (CA) and Phoenix (AZ)
  - Very little data on raining & snowing weather

⚠️ Lots of biases in data! ⚠️

Types of sampling

- Non-probability sampling
- Random sampling
  - Simple random sampling
  - Stratified sampling
  - Weighted sampling
  - Importance sampling
  - Reservoir sampling
  - ...

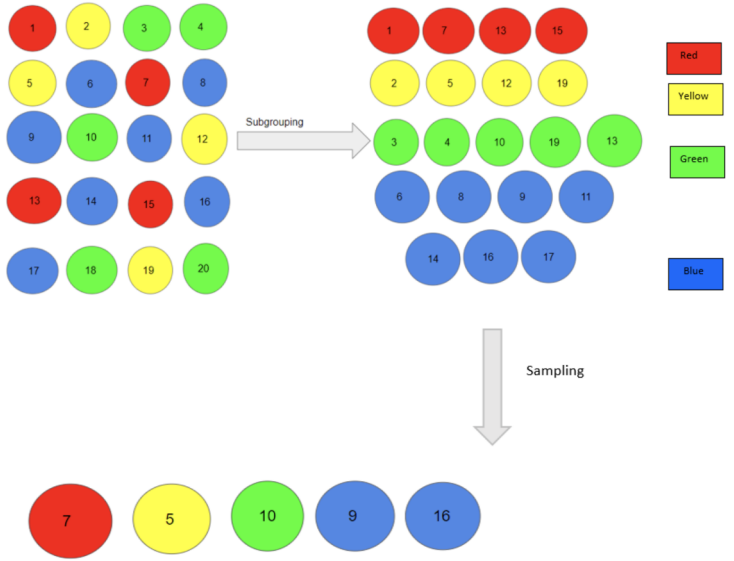
Simple random sampling

- Each sample in population has an equal chance of being selected
  - E.g. select 10% of all samples in population

Pros	Cons
• Simple (easiest type of random sampling)	• No representation guarantee: might exclude rare classes (black swan!)

Stratified sampling

- Divide population by subgroups
  - Slices of data
    - 20% of each age group: 18-24, 25-34, 35+, etc.
  - Classes
    - 2% of each class



Pros	Cons
Minor groups are represented	Can't be used when: ● samples can't be put into subgroups ● samples can belong in multiple subgroups (multilabel)

Image from https://www.analyticsvidhya.com/blog/2019/09/data-scientists-guide-8-types-of-sampling-techniques/

Weighted sampling

- Each element is given a weight, which determines the probability of being selected.
  - If you want to select a sample 30% of the time, give it 3/10 weight
- Might embed domain knowledge
  - E.g. know distribution of your target population or want to prioritize recent samples

```
random.choices(population=[1, 2, 3, 4, 100, 1000],
                           weights=[0.2, 0.2, 0.2, 0.2, 0.1, 0.1],
               k=2)
```

```
random.choices(population=[1, 1, 2, 2, 3, 3, 4, 4, 100, 1000],
               k=2)
```

Importance sampling

- Useful when sampling from P(x) is expensive, slow, or infeasible
  - Sample x ~ Q(x) ← easier to sample from
  - Weight by P(x)/Q(x)
- ⚠ Calculating P(x)/Q(x) might be expensive ⚠

$$E_{P(x)}[x] = \sum_x P(x)\, x = \sum_x Q(x)\, x\, \frac{P(x)}{Q(x)} = E_{Q(x)}[x\frac{P(x)}{Q(x)}]$$

Importance sampling

- Useful when sampling from P(x) is expensive, slow, or infeasible
  - Sample x ~ Q(x) ← easier to sample from
  - Weight by P(x)/Q(x)
- ⚠ Calculating P(x)/Q(x) might be expensive ⚠
- E.g. essential for RL
  - Too expensive to estimate reward under new policy, so use old policy

$$E_{P(x)}[x] = \sum_x P(x)\, x = \sum_x Q(x)\, x\, \frac{P(x)}{Q(x)} = E_{Q(x)}[x\frac{P(x)}{Q(x)}]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim\bar{\pi}_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\,(\mathbf{a}_t|\mathbf{s}_t)\left(\prod_{t'=1}^{t}\frac{\pi_\theta\,(\mathbf{a}_{t'}|\mathbf{s}_{t'})}{\bar{\pi}_\theta(\mathbf{a}_{t'}|\mathbf{s}_{t'})}\right)\left(\sum_{t'=t}^{T} r(\mathbf{s}_{t'},\mathbf{a}_{t'})\right)\right]$$

use old policy to sample data

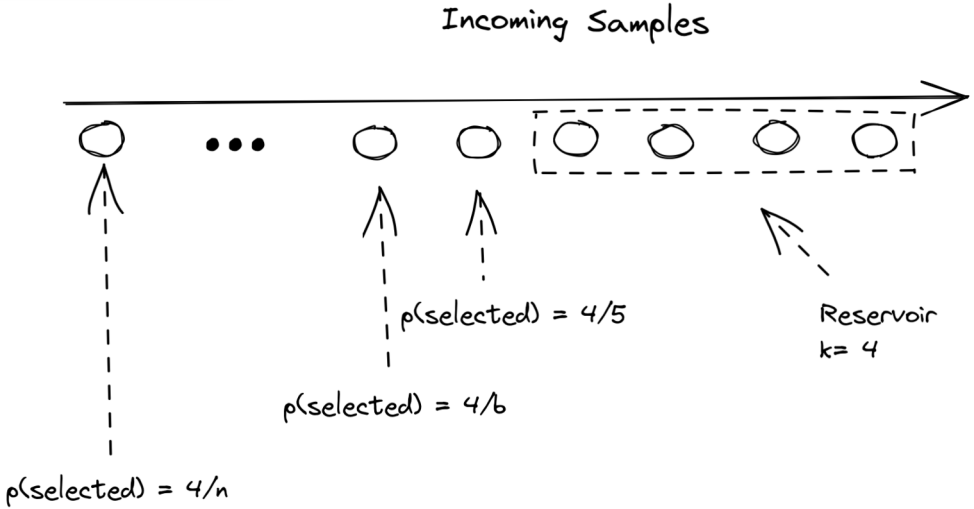
old policy

Reservoir sampling: problem

- Need select k samples from a stream of n samples with equal probability
  - n is unknown
  - impossible/inefficient to fit all in memory
- Can stop the stream any moment and get the required samples

Reservoir sampling: solution

1. First k elements are put in reservoir
2. For each incoming i^{th} element, generate a random number j between 1 and i
    a. If $1 \leq j \leq k$: replace j^{th} in reservoir with i^{th}
3. Each incoming element has k/i chance of being in reservoir!



Also checkout algorithm L (based on geometric distribution)

With vs. without replacement

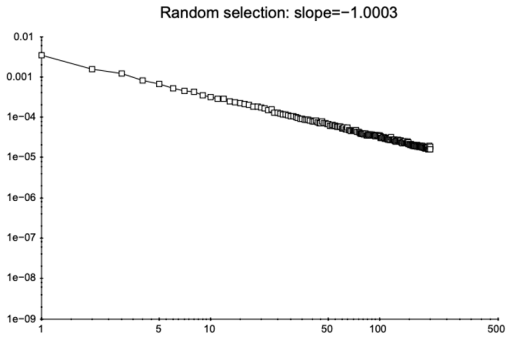
With replacement	**Without replacement**
Same item can be chosen more than once	Same item can’t be chosen more than once
• No covariance between two chosen samples • Approximate true population distribution	• Covariance between two chosen samples • Covariance reduced as dataset size becomes large
Bagging	Mini-batch gradient descent

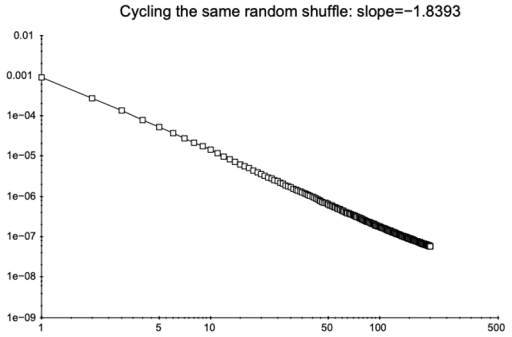
Why do we use epochs instead of just sampling with replacement from the entire dataset?

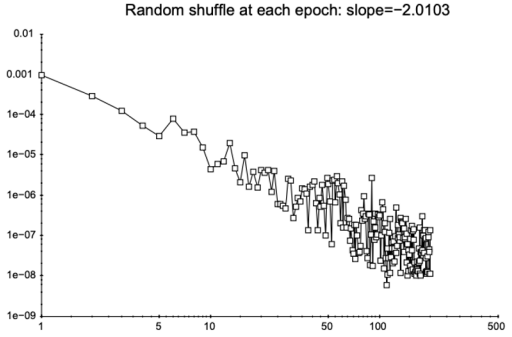
With vs. without replacements

Because empirically it's converged faster

proven for strongly convex loss functions



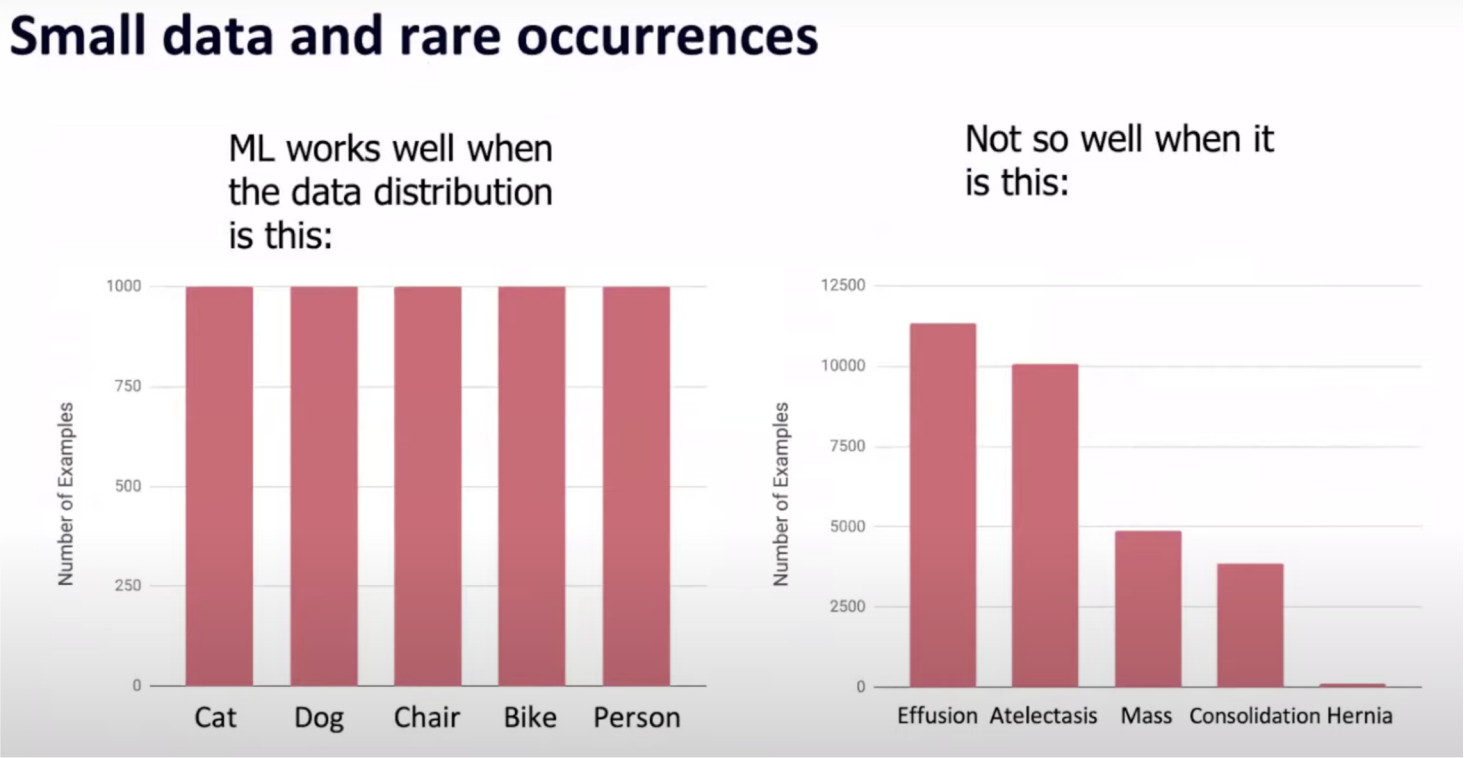




Curiously Fast Convergence of some Stochastic Gradient Descent Algorithms (Leon Bottou, 2009)
Why Random Reshuffling Beats Stochastic Gradient Descent (Gurbuzbalaba et al., 2015)

5. Class imbalance

Class imbalance



Andrew Ng: Bridging AI's Proof-of-Concept to Production Gap (2020)

Why is class imbalance hard?

- Not enough signal to learn about rare classes

Why is class imbalance hard?

- Not enough signal to learn about rare classes
- Statistically, predicting majority label has higher chance of being right
  - If a majority class accounts 99% of data, always predicting it gives 99% accuracy

Why is class imbalance hard?

- Not enough signal to learn about rare classes
- Statistically, predicting majority label has higher chance of being right
- Asymmetric cost of errors: different cost of wrong predictions

Asymmetric cost of errors: regression

- 95th percentile: $10K
- Median: $250

Thanks Eugene Yan for this example!

Asymmetric cost of errors: regression

100% error difference

Not OK

- $10K bill: off by $10K
- $250 bill: off by $250

OK

Thanks Eugene Yan for this example!

1. Data augmentation

"Data augmentation is the new feature engineering"

- Josh Wills, prev Director of Data Engineering @ Slack

Data augmentation: goals

- Improve model’s performance overall or on certain classes
- Generalize better
- Enforce certain behaviors

Data augmentation

1. Simple label-preserving transformation
2. Perturbation
3. Data synthesis

Label-preserving: Computer Vision

Random cropping, flipping, erasing, etc.

Image from [An Efficient Multi-Scale Focusing Attention Network for Person Re-Identification] (Huang et al., 2021)

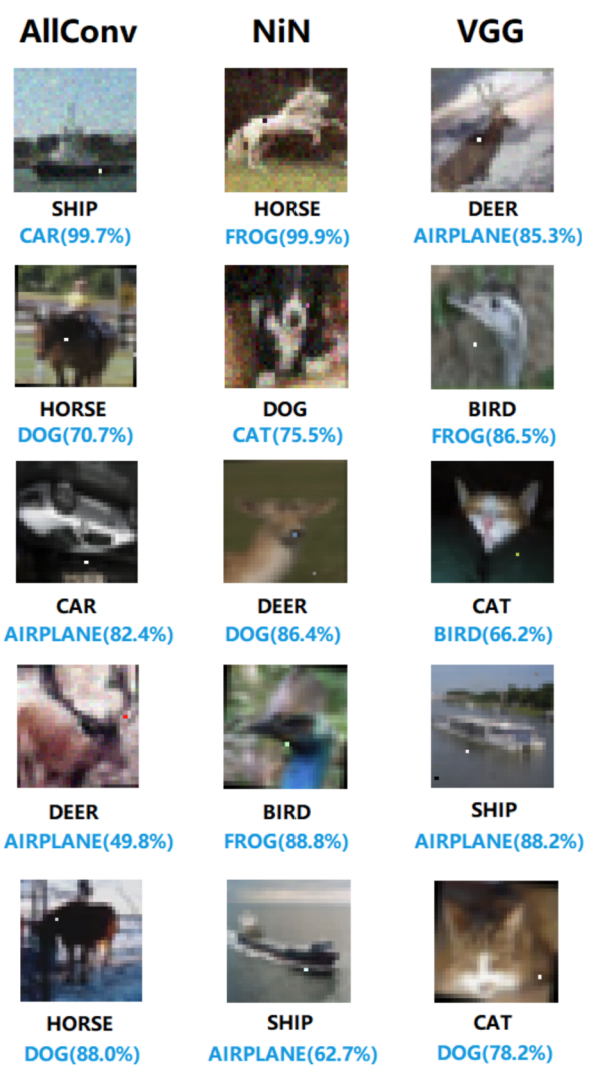
Label-preserving: NLP

Original sentences	I'm so happy to see you.
Generated sentences	I'm so **glad** to see you. I'm so happy to see **y'all**. I'm **very** happy to see you.

Perturbation: neural networks can be sensitive to noise

- 67.97% Kaggle CIFAR-10 test images
- 16.04% ImageNet test images

can be misclassified by changing just one pixel (Su et al., 2017)

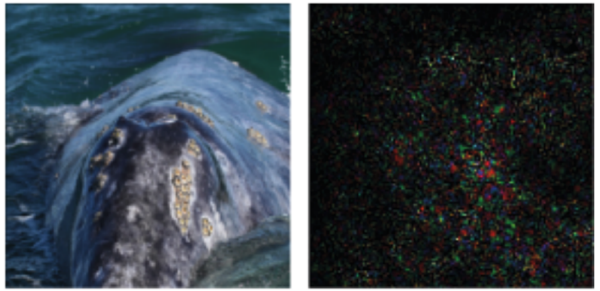


Perturbation: Computer Vision

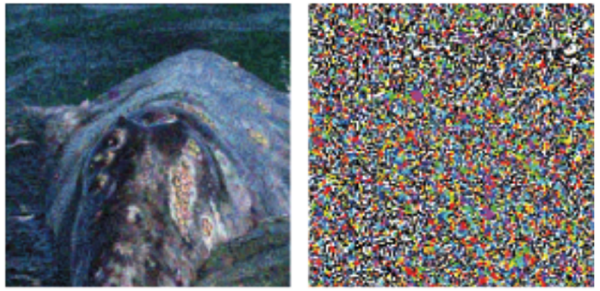
- Random noise
- Search strategy
  - DeepFool (Moosavi-Dezfooli et al., 2016): find the minimal noise injection needed to cause a misclassification with high confidence.

Whale

Turtle
noise by DeepFool



Turtle
noise by fast gradient sign



Perturbation: NLP

- Random replacement
  - e.g. BERT (10% * 15% = 1.5%)

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., 2018)

Data synthesis: NLP

- Template-based
  - Very common in conversational AI
- Language model-based

Template	Find me a [CUISINE] restaurant within [NUMBER] miles of [LOCATION].
Generated queries	• Find me a **Vietnamese** restaurant within **2** miles of **my office**. • Find me a **Thai** restaurant within **5** miles of **my home.** • Find me a **Mexican** restaurant within **3** miles of **Google headquarters**.

Data Synthesis: Computer Vision

- Mixup
  - Create convex combination of samples of different classes
    - Labels: cat [3], dog [4]
    - Mixup: 30% dog, 70% cat [0.3 * 3 + 0.7 * 4 = 3.7]

mixup: Beyond Empirical Risk Minimization (Zhang et al., 2017)

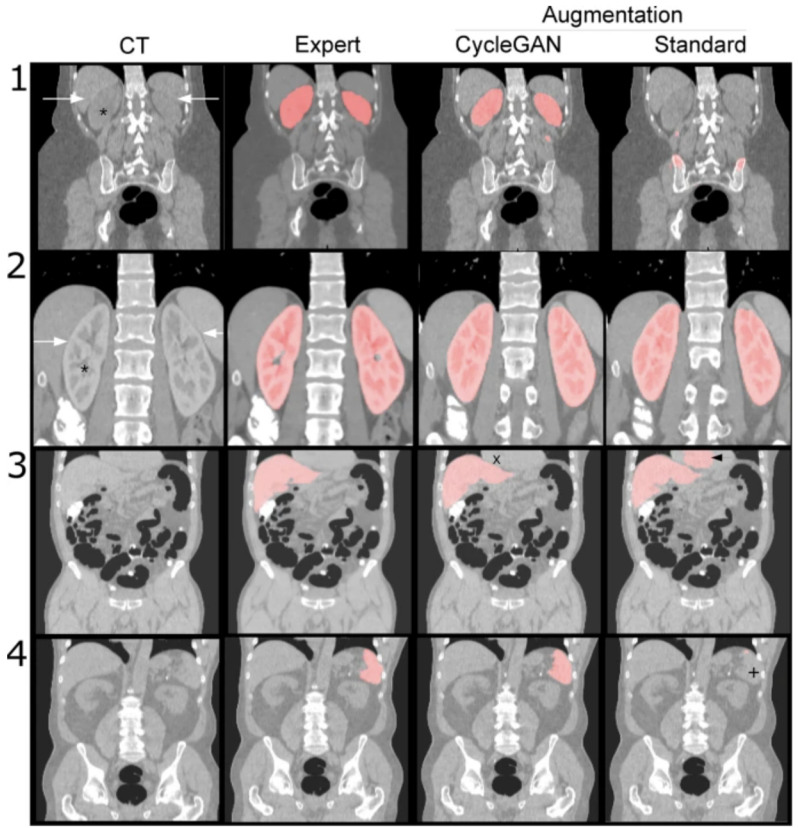
Data Synthesis: Computer Vision

- Mixup
  - Incentivize models to learn linear relationships
  - Improves generalization on speech and tabular data
  - Can be used to stabilize the training of GANs

https://forums.fast.ai/t/mixup-data-augmentation/22764

Data augmentation: GAN

Example: kidney segmentation with data augmentation by CycleGAN



Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks (Sandfort et al., 2019)

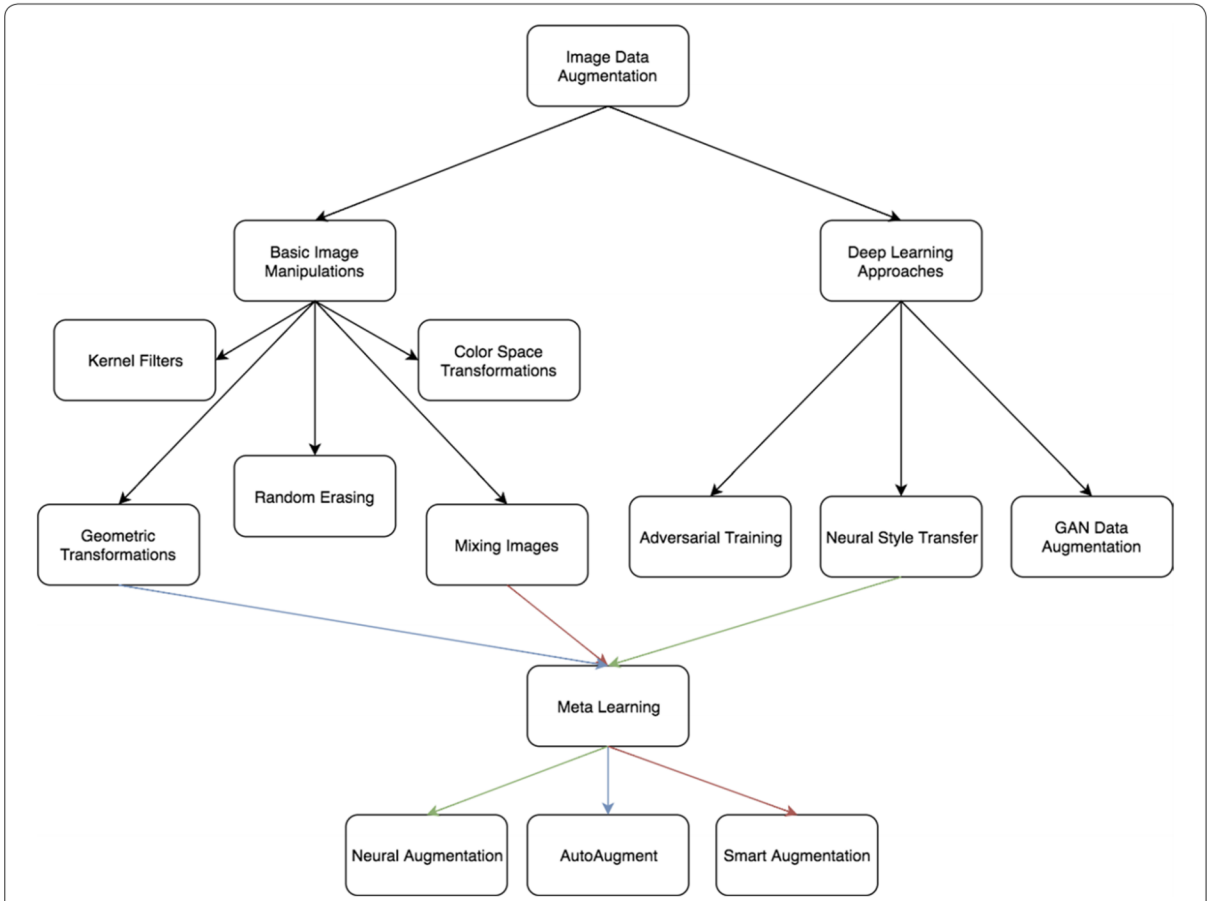


Fig. 2 A taxonomy of image data augmentations covered; the colored lines in the figure depict which data augmentation method the corresponding meta-learning scheme uses, for example, meta-learning using Neural Style Transfer is covered in neural augmentation [36]

A survey on Image Data Augmentation for Deep Learning (Connor Shorten & Taghi M. Khoshgoftaar, 2019)

Learned features
vs.
engineered features

Feature engineering

Wait, doesn’t deep learning promise no more feature engineering?

Feature engineering

Wait, doesn’t deep learning promise no more feature engineering?

- We’re still very far from that point
- Many ML models in industry aren’t deep learning

Engineered features: text

Stopword removal I have a dog. He's sleeping.

↓

Lemmatization I have dog. He's sleeping.

↓

Contraction I have dog. He's sleep.

↓

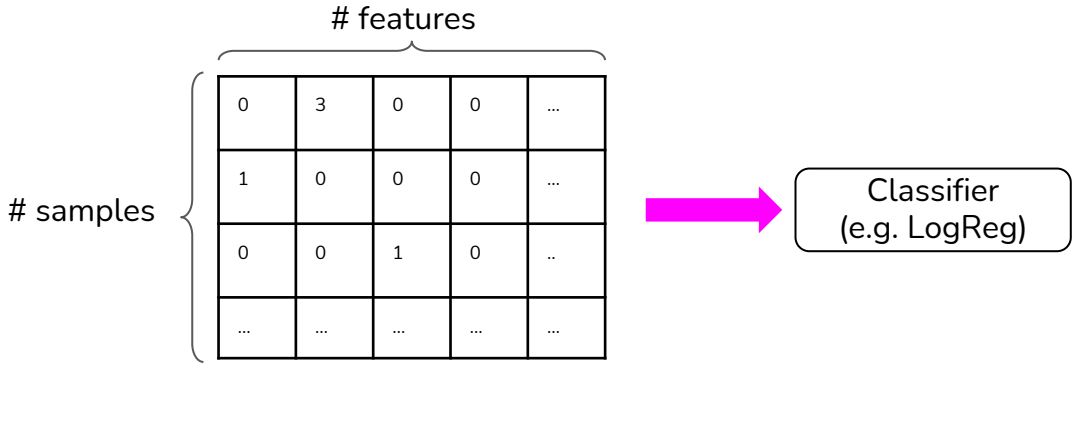
Punctuation I have dog. He is sleep.

↓

Lowercase I have dog He is sleep

↓

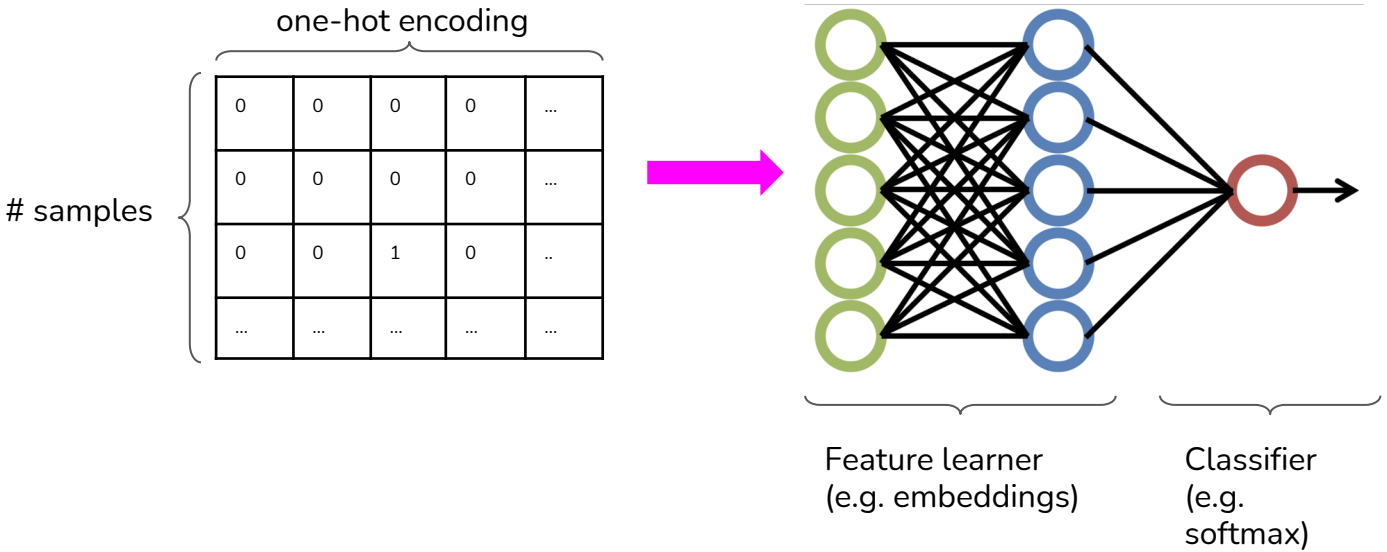
N-gram i have dog he is sleep



Features

I	you	have	dog	cat	he	she	is	they	sleep	I, have	have, dog	good, dog	...
1	0	1	1	0	1	0	1	0	1	1	1	0	...

Learned features: text



Text I have a dog. He's sleeping.

One-hot

I	you	have	dog	cat	he	she	is	they	sleep	mom	food	yes	...
1	0	1	1	0	1	0	1	0	0	0	0	0	...

Learned features: spam classification

Comment ID	Time	User	Text	# ▲	# ▼	Link	# img	Thread ID	Reply to	# replies	...
93880839	2020-10-30 T 10:45 UTC	gitrekt	Your mom is a nice lady.	1	0	0	0	2332332	n0tab0t	1	...

User ID	Created	User	Subs	# ▲	# ▼	# replies	Karma	# threads	Verified email	Awards	...
4402903	2015-01-57 T 3:09 PST	gitrekt	[r/ml, r/memes, r/socialist]	15	90	28	304	776	No		...

Thread ID	Time	User	Text	# ▲	# ▼	Link	# img	# replies	# views	Awards	...
93883208	2020-10-30 T 2:45 PST	doge	Human is temporary, AGI is forever	120	50	1	0	32	2405	1	...

Feature engineering: spam classification

Even more features:

- Post frequency, max posts per day
- Post repetitiveness
- Language detection, typos, abnormal punctuations, ratio uppercase/lowercase
- IP, other users from the same IP
- NSFW words, blacklisted links
- Targeted users
- ...

Feature engineering

- For complex tasks, number of features can go up to millions!
- Lots of ML production work involves coming up with new features
  - Fraudsters come up with new techniques very fast, so need to come up with new features very fast to counter
- Often require subject matter expertise

Feature engineering operations

Kinbert Chou

Common feature engineering ops

1. Handling missing values
2. Scaling
3. Discretization
4. Categorical features
5. Feature crossing
6. Positional embeddings

Handling missing values

- Not all missing values are equal
  - Missing not at random (MNAR)
  - Missing at random (MAR)
  - Missing completely at random (MCAR)

Handling missing values

Missing not at random – when a value is missing due to the value itself

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	($350,000?)		2	Engineer	Yes
5	35	B	($350,000?)	Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

Missing at random – when a value is missing due to another observed variable

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

Missing completely at random – there is no pattern to which values are missing

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Deletion – removing data with missing entries
- Imputation – filling missing fields with certain values

Many people prefer deletion not because it’s better, but it’s easier to do

Handling missing values

- Deletion
  - Column deletion – remove columns with too many missing entries
    - drawbacks – even if half the values are missing, the remaining data still potentially useful information for predictions
    - e.g. even if over half the column for ‘Marital status’ is missing, marital status is still highly correlated with house purchasing
  - Row deletion

Marital status
Married
Single
Single

Handling missing values

- Deletion
  - Column deletion
  - Row deletion

Handling missing values

- Row deletion
  - Good for: data missing completely at random (MCAR) and few values missing

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1	39	A	150,000	Married	1	Engineer	No
2	27	B	50,000	Single	0	Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	75,000	Married	2	Engineer	Yes
5	35	B	35,000	Single	0	Doctor	Yes
6	32	A	50,000	Married	0	Teacher	No
7	33	B	60,000	Single	2	Teacher	No
8	20	B	10,000	Single	1	Student	No

Handling missing values

- Row deletion
  - Bad when many examples have missing fields

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Row deletion
  - Bad for: missing values are not at random (MNAR)
  - Missing information is information itself

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B	($350,000?)		2	Engineer	Yes
5	35	B	($350,000?)	Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Handling missing values

- Row deletion
  - Bad for: missing data at random (MAR)
  - Can potentially bias data – we've accidentally removed all examples with gender 'A'

ID	Age	Gender	Annual income	Marital status	Number of children	Job	Buy?
1		A	150,000		1	Engineer	No
2	27	B	50,000			Teacher	No
3		A	100,000	Married	2		Yes
4	40	B			2	Engineer	Yes
5	35	B		Single	0	Doctor	Yes
6		A	50,000		0	Teacher	No
7	33	B	60,000	Single		Teacher	No
8	20	B	10,000			Student	No

Imputation

- Fill missing fields with certain values
  - Defaults
    - E.g. 0, or the empty string, etc.
  - Statistical measures – mean, median, mode
    - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July

Imputation

- Fill missing fields with certain values
  - Defaults
    - E.g. 0, or the empty string, etc.
  - Statistical measures – mean, median, mode
    - e.g. if a day in July is missing its temperature value, fill it with the median temperature in July

Avoid filling missing values with possible values!

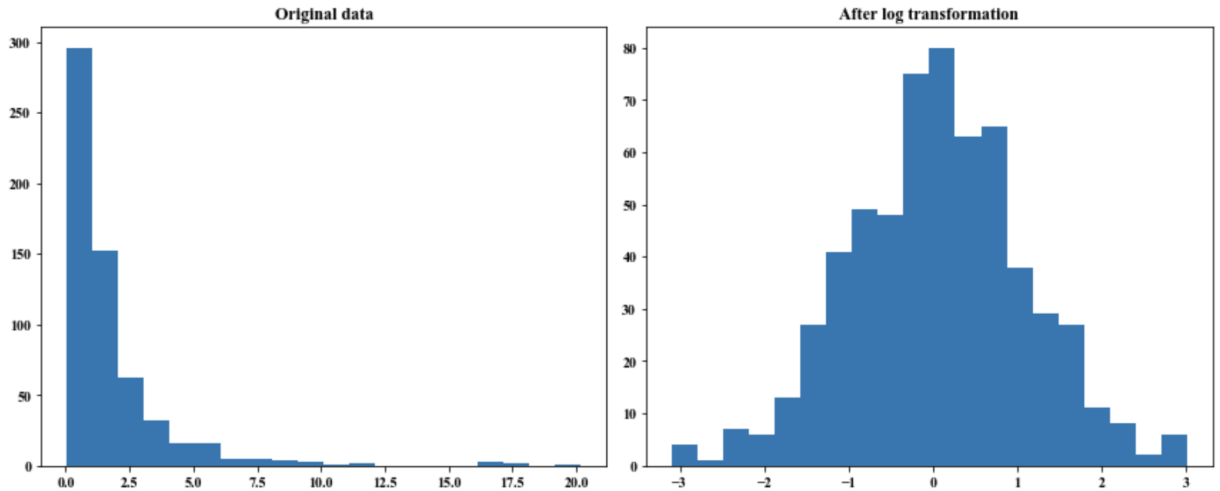
Scaling

Types of scaling

scaling type	**use case**
min/max normalization	Any -- no assumptions about variables
z-score normalization	When variables follow a normal distribution
log scaling	When variables follow an exponential distribution

Log scaling

- Help with skewed data
- Often gives performance gain



Scaling

scaling type	use case
min/max normalization	Any -- no assumptions about variables
z-score normalization	When variables follow a normal distribution
log scaling	When variables follow an exponential distribution

- scaling can be a common source of data leakage

Scaling

scaling type	use case
min/max normalization	Any -- no assumptions about variables
z-score normalization	When variables follow a normal distribution
log scaling	When variables follow an exponential distribution

- scaling can be a common source of data leakage
- scaling variables requires global statistics

Discretization

- Turning a continuous feature into a discrete feature (quantization)

Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
    - Incorporate knowledge/expertise about each variable by constructing specific buckets

Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
  - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
  - Income
    - Lower income: x < $35,000
    - Middle income: $35,000 < x < $100,000
    - High income: x > $100,000

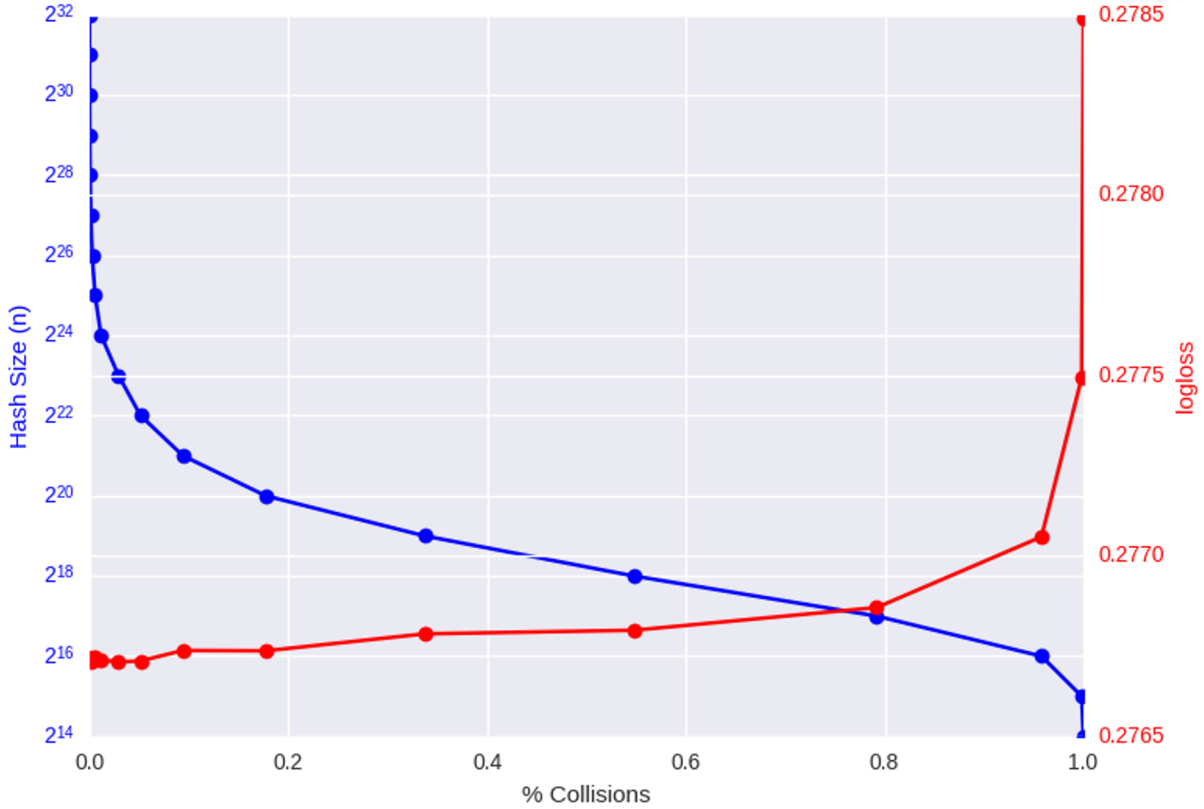
Discretization

- Turning a continuous feature into a discrete feature (quantization)
- Create buckets for different ranges
  - Incorporate knowledge/expertise about each variable by constructing specific buckets
- Examples
  - Income
    - Lower income: x < $35,000
    - Middle income: $35,000 <= x < $100,000
    - High income: x >= $100,000
  - Age
    - Minors: x < 18
    - College: 18 <= x < 22
    - Young adult: 22 <= x < 30
    - 30 <= x < 40
    - 40 <= x < 65
    - Seniors: x >= 65

Encoding Categorical Features

- Hashing – use a hash function to hash categories to different indexes
  - e.g. hash(“Nike”) = 0, hash(“Adidas”) = 27, etc...
- Benefits – you can choose how large the hash space is
- Drawbacks – two categories being hashed to the same index

Encoding Categorical Features



Encoding Categorical Features

- Choose a hash space large enough to reduce collisions
- Choose functions with properties beneficial to your use case
    - Locality-sensitive hashing

Hashing Trick Takeaways

- Hashing trick considered “hacky” by academics
- Widely used in industry and in machine learning frameworks
- Useful in practice for continual learning in production

Feature Crossing

- Combine two or more features to create a new feature

Marriage	Single	Married	Single	Single	Married
Children	0	2	1	0	1
Marriage & children	Single, 0	Married, 2	Single, 1	Single, 0	Married, 1

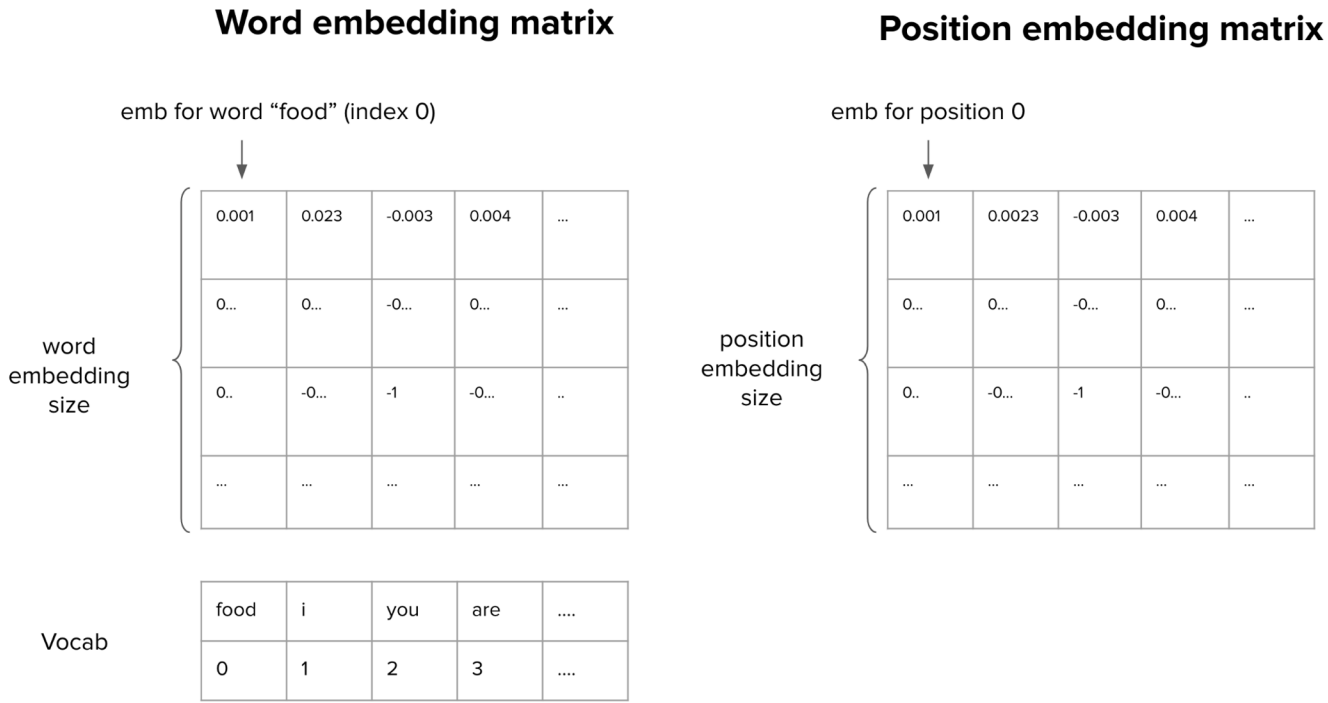
Feature Crossing

- Helps models learn non-linear relationships between variables
- Warning – feature crossing can blow up your feature space
  - e.g. Feature A and B both have 100 categories → Feature A x B will have 10,000 categories
  - Need even more data to learn this new feature space
  - Blowing up feature space can increase risk of overfitting

Very common in RecSys & CTR with models like DeepFM and xDeepFM

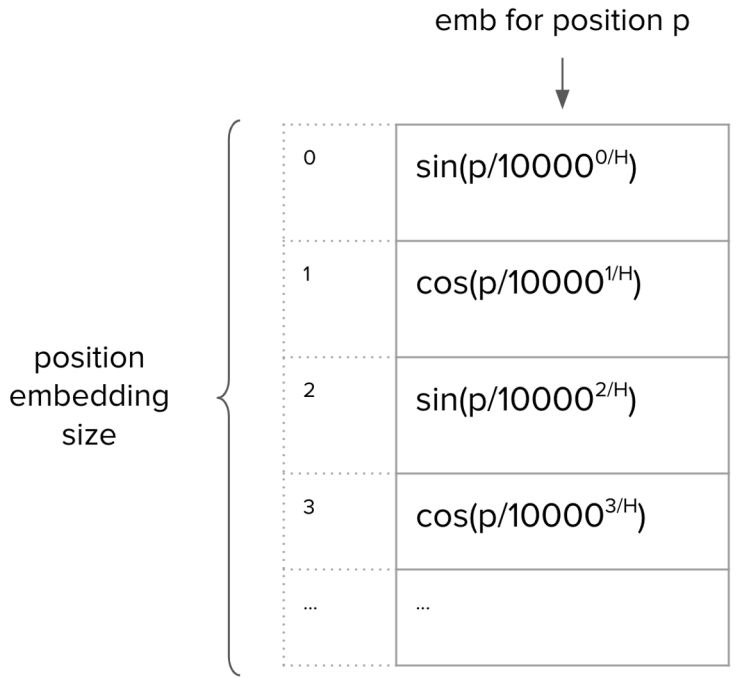
Positional Embeddings

- Popularized in *Attention is All You Need* paper
- Similar to word embeddings
  - Can be either learned or fixed



Positional Embeddings

- Fourier features



Data leakage

Data leakage

- Some form of the label “leaks” into the features
- This same information is not available during inference

Data leakage: example 1

- Problem: detect lung cancer from CT scans
- Data: collected from hospital A
- Performs well on test data from hospital A
- Performs poorly on test data from hospital B

Patient ID	Date	Doctor note	Medical record	Scanner type	CT scan

Data leakage: example 1

- Problem: detect lung cancer from CT scans
- Data: collected from hospital A
- Performs well on test data from hospital A
- Performs poorly on test data from hospital B

Patient ID	Date	Doctor note	Medical record	Scanner type	CT scan

At hospital A, when doctors suspect that a patient has lung cancer, they send that patient to a higher-quality scanner

Data leakage: example 2

- Problem: predicting how many views an article will get
- Data: historical data on the site
- Where might data leakage come from?

Article ID	Date	Title	Article	Author	Language	Translations

Data leakage: example 2

- Problem: predicting how many views an article will get
- Data: historical data on the site

Not leakage because author popularity also available during inference

The site only translate articles that are already gaining attention

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time

Partition: shuffle then split

	Week 1	Week 2	Week 3	Week 4	Week 5
Test split	X11	X21	X31	X41	X51
Valid split	X12	X22	X32	X42	X52
Train split	X13	X23	X33	X43	X53
	X14	X24	X34	X44	X54
	...	...	...	...	...

Aim for similar distributions of labels across splits
e.g. each split has 90% NEGATIVE, 10% POSITIVE

Partition: shuffle then split

	Week 1	Week 2	Week 3	Week 4	Week 5
Test split	X11			X41	X51
Valid split	X12			X42	X52
Train split	X13			X43	X53
	X14			X44	X54
	...	...	...	...	...

⚠️ Not representative of real-world usage! ⚠️

Partition: shuffle then split

	Week 1	Week 2	Week 3	Week 4	Week 5
Test split	X11			X41	X51
Valid split	X12			X42	X52
Train split	X13			X43	X53
	X14			X44	X54
	...	...	...	...	...

A source of data leakage. Examples:

- stock price prediction
- song recommendation

A better partition

Train split					
Week 1	Week 2	Week 3	Week 4	Week 5	
X11	X21	X31	X41	X51	**Valid split**
X12	X22	X32	X42	X52	
X13	X23	X33	X43	X53	**Test split**
X14	X24	X34	X44	X54	
...	...	...	...	...	

Solution: split data by time

			Train split		
Week 1	Week 2	Week 3	Week 4	Week 5	
X11	X21	X31	X41	X51	**Valid split**
X12	X22	X32	X42	X52	
X13	X23	X33	X43	X53	**Test split**
X14	X24	X34	X44	X54	
...	...	...	...	...	

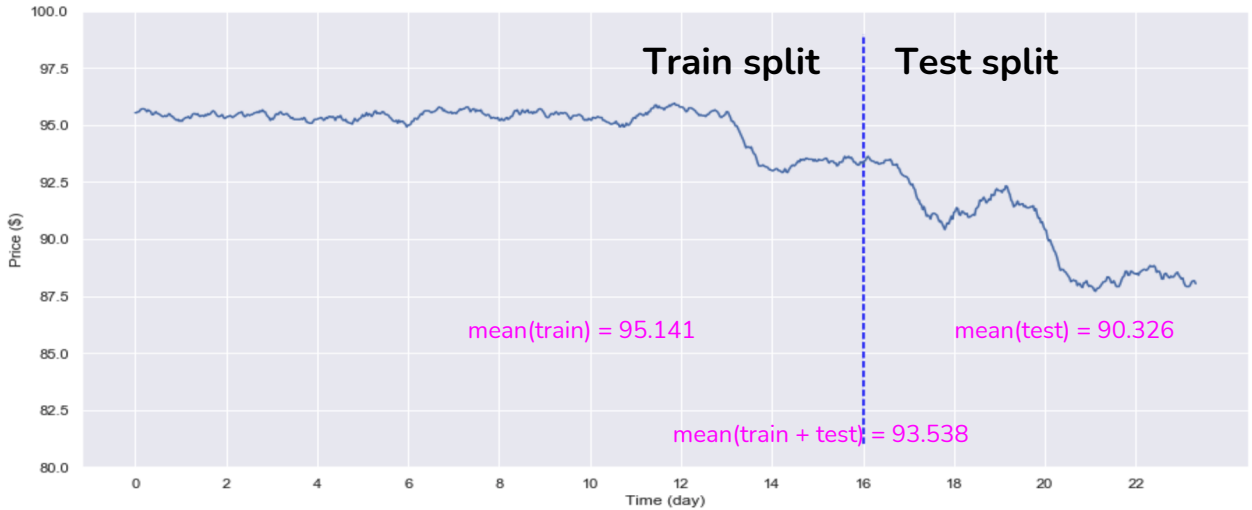
Also forces you to think about the cold-start problem

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
    a. Use the whole dataset (including valid/test) to generate global statistics/info

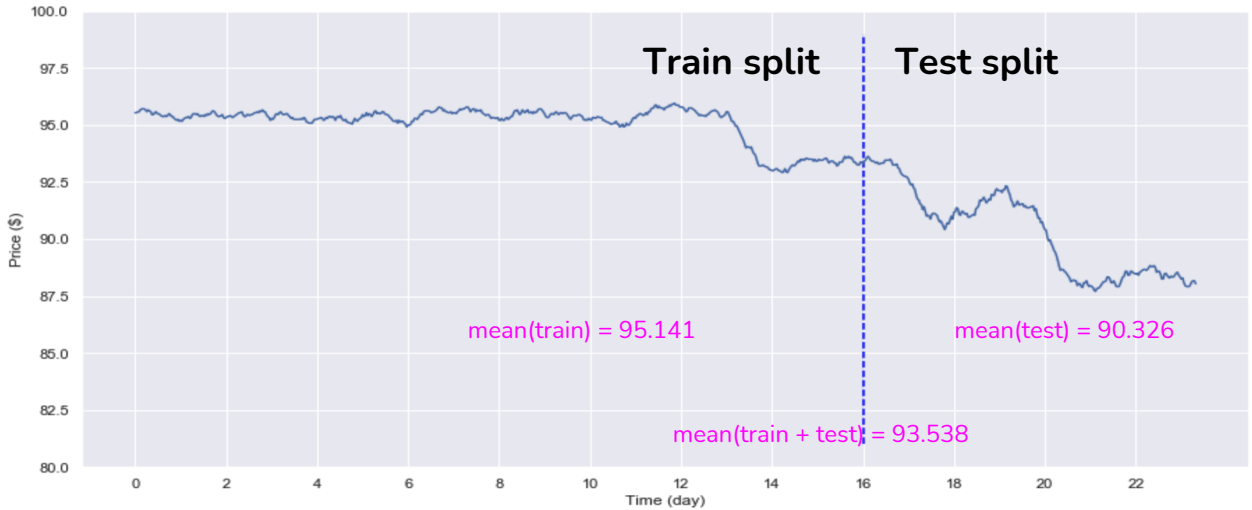
2. Data processing before splitting

- Use the whole dataset (including valid/test) to generate global statistics/info
  - mean, variance, min, max, n-gram count, vocabulary, etc.
- Statistics are then used to process test data
  - scale, fill in missing values, etc.



2. Data processing before splitting

- Use the whole dataset (including valid/test) to generate global statistics/info
- Solution:
  - Split your data before scaling/filling in missing values
  - Split even before any EDA to ensure you're blind to the test set

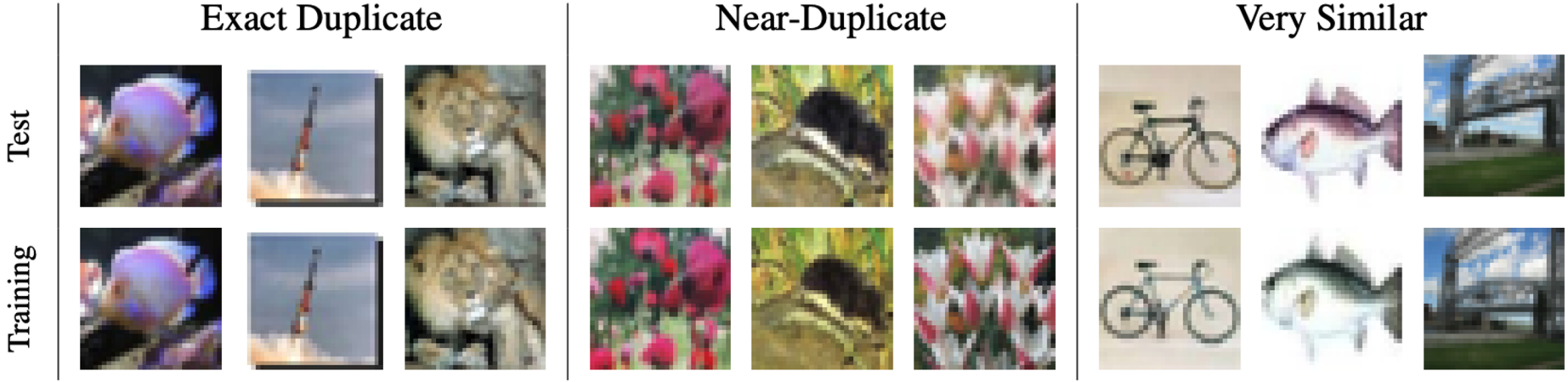


Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
3. Poor handling of data duplication before splitting
    a. Test set includes data from the train set

3. Poor handling of data duplication before splitting

- Datasets come with duplicates & near-duplicates
  - 3.3% CIFAR-10 and 10% CIFAR-100 test images have dups in training set
  - Removing dups increases errors 17.05% -> 19.38% on CIFAR-100 [PyramidNet-272-200}



Do we train on test data? Purging CIFAR of near-duplicates (Barz & Denzler, 2019)

3. Poor handling of data duplication before splitting

- Datasets come with duplicates & near-duplicates
- Oversampling can cause duplications

3. Poor handling of data duplication before splitting

- Test set includes data from the train set
- Solution:
  - Deduplicate data before splitting
  - Oversample after splitting

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
3. Poor handling of data duplication before splitting
4. Group leakage
    a. A group of examples have strongly correlated labels but are divided into different splits

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
3. Poor handling of data duplication before splitting
4. Group leakage
    a. A group of examples have strongly correlated labels but are divided into different splits
    b. Example: CT scans of the same patient a week apart
    c. Solution: Understand your data and keep track of its metadata

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
3. Poor handling of data duplication before splitting
4. Group leakage
5. Leakage from data generation & collection process
    a. Example: doctors send high-risk patients to a better scanner
    b. Solution: Data normalization + subject matter expertise

Causes of data leakage

1. Splitting time-correlated data randomly instead of by time
2. Data processing before splitting
3. Poor handling of data duplication before splitting
4. Group leakage
5. Leakage from data generation & collection process

How to detect leakage?

1. Measure correlation of a feature with labels
    a. A feature alone might not cause leakage, but 2 features together might

How to detect leakage?

1. Measure correlation of a feature with labels
2. Feature ablation study
    a. If removing a feature causes the model performance to decrease significantly, figure out why.

How to detect leakage?

1. Measure correlation of a feature with labels
2. Feature ablation study
3. Monitor model performance as more features are added
    a. Sudden increase: either a very good feature or leakage!

How to engineer good features

Evaluating a feature

1. Feature importance
2. Feature generalization

Measuring a feature’s importance

How much the model performance deteriorates
if a feature or a set of features containing that feature
is removed from the model?

Measuring a feature's importance

- XGBoost

get_score(*fmap='', importance_type='weight'*)

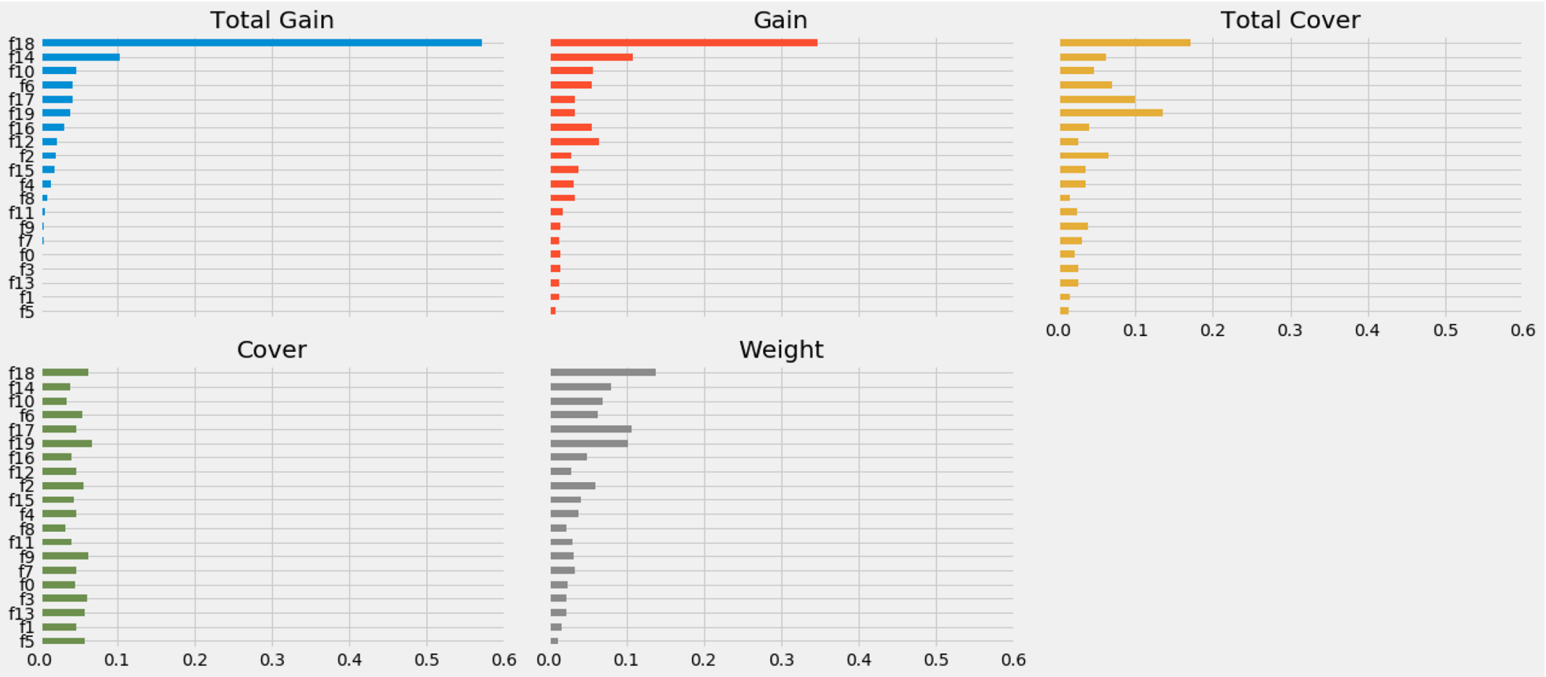
Get feature importance of each feature. For tree model Importance type can be defined as:

- 'weight': the number of times a feature is used to split the data across all trees.
- 'gain': the average gain across all splits the feature is used in.
- 'cover': the average coverage across all splits the feature is used in.
- 'total_gain': the total gain across all splits the feature is used in.
- 'total_cover': the total coverage across all splits the feature is used in.

https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.Booster.get_score

Measuring a feature's importance

- XGBoost



[The Multiple faces of 'Feature importance' in XGBoost](#) (Amjad Abu-Rmileh, 2019)

SHAP: SHapley Additive exPlanations

- Measuring a feature's contribution to a single prediction

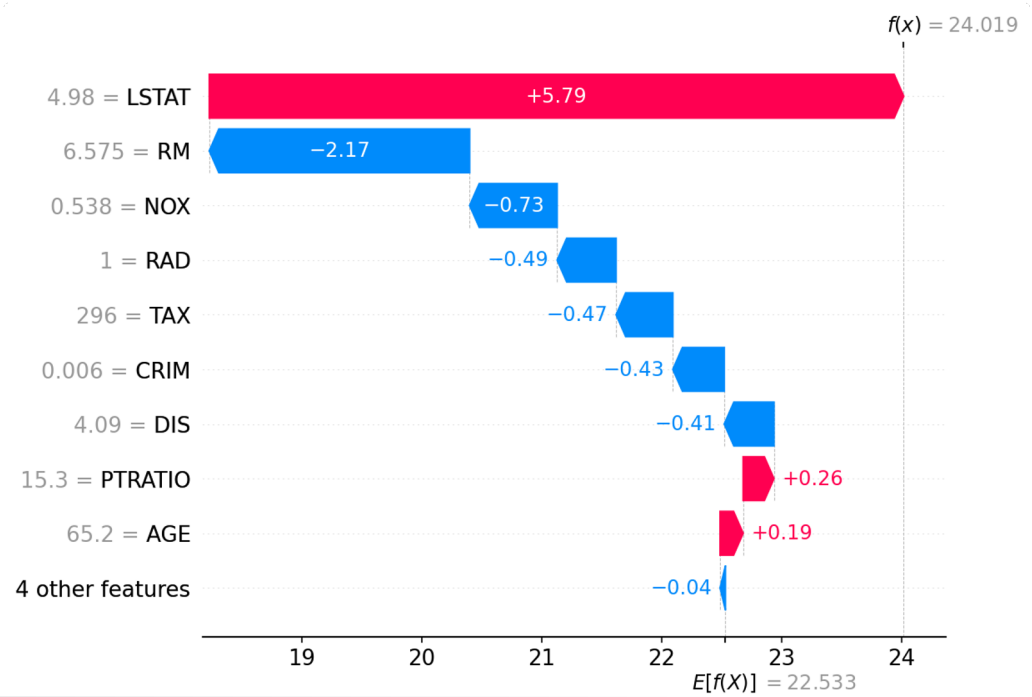


Image by Scott Lundberg from https://github.com/slundberg/shap

SHAP: SHapley Additive exPlanations

- Measuring a feature's contribution to the entire model

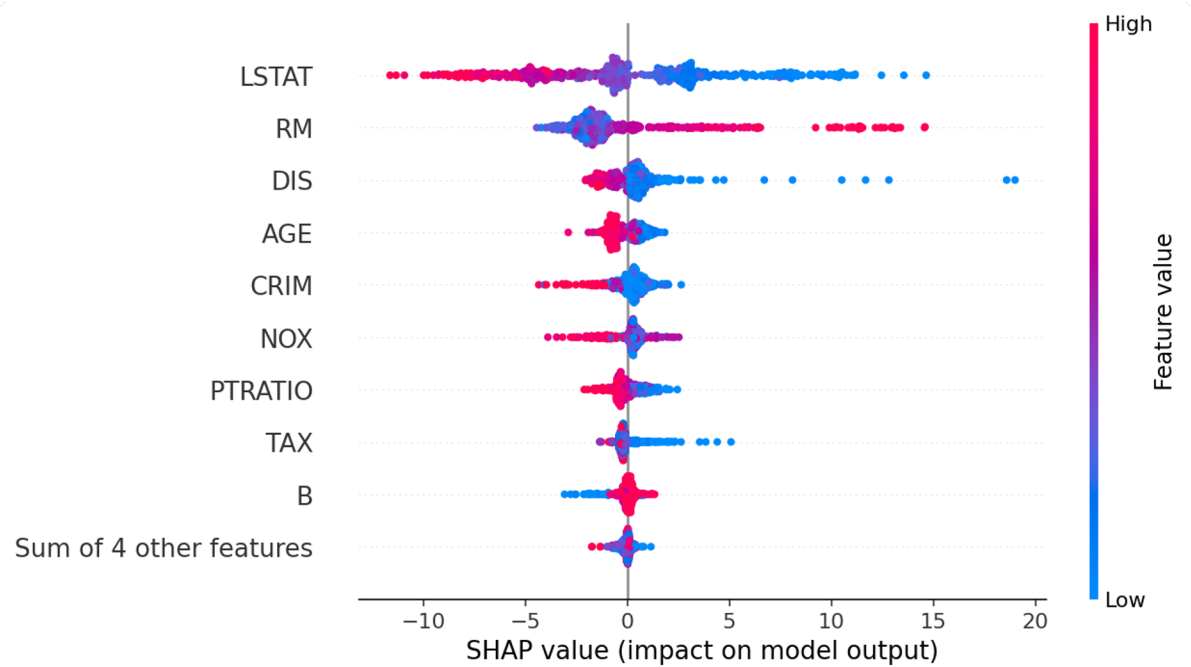
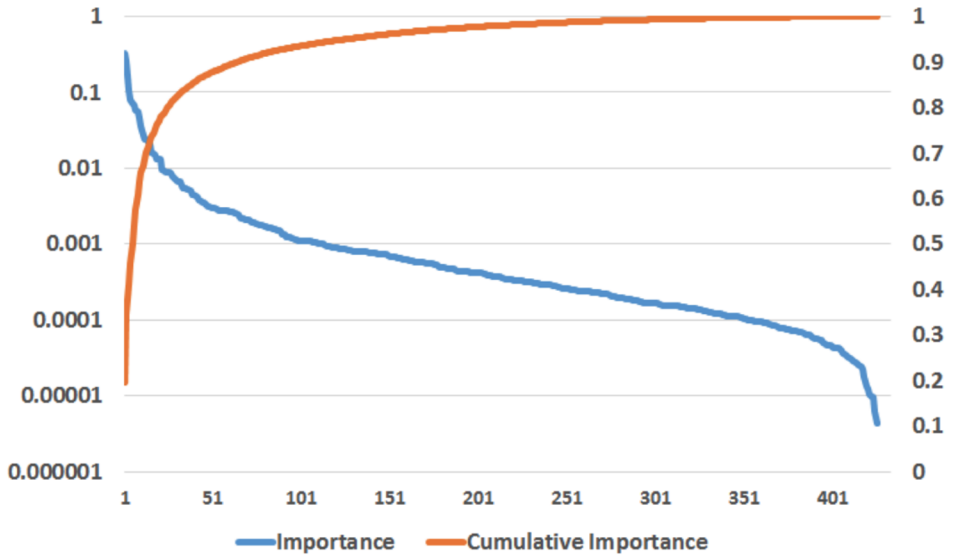


Image by Scott Lundberg from https://github.com/slundberg/shap

Measuring feature importance @ Facebook

- Top 10 features: 50% total feature importance
- Bottom 300 features: <1% total feature importance



Practical Lessons from Predicting Clicks on Ads at Facebook (He et al., 2014)

https://github.com/interpretml/interpret

```
ebm = ExplainableBoostingClassifier()
```

Feature engineering: the more the better?

- Adding more features tends to improve model performance

How can having too many features be bad?

Too many features can be bad ...

- Training:
  - Overfitting
  - More features, more opportunity for data leakage

Too many features can be bad ...

- Training:
  - Overfitting
  - More features, more opportunity for data leakage
- Inference
  - Increase inference latency with online prediction
  - Might cause increased memory usage -> more expensive instance required

Too many features can be bad ...

- Training:
  - Overfitting
  - More features, more opportunity for data leakage
- Inference
  - Increase inference latency with online prediction
  - Might cause increased memory usage -> more expensive instance required
- Stale features become technical debts
  - E.g. if zip codes are no longer allowed for predictions, all features that use zip codes will need to be updated

Too many features can be bad ...

- Training:
  - Overfitting
  - More features, more opportunity for data leakage
- Inference
  - Increase inference latency with online prediction
  - Might cause increased memory usage -> more expensive instance required
- Stale features become technical debts

Solution:

- Clean up stale / ineffective features
- Store features in case you want to reuse them
  - Feature management

9 best practices for feature engineering

1. Split data by time instead of doing it randomly.
2. If you oversample your data, do it after splitting.
3. Use statistics/info from the train split, instead of the entire data, for feature engineering: scaling, normalizing, handling missing values, creating n-gram count, item encoding, etc.
4. Understand how your data is generated, collected, and processed. Involve domain experts if necessary.
5. Keep track of data lineage.
6. Understand feature importance to your model.
7. Measure correlation between features and labels.
8. Use features that generalize well.
9. Remove stale features from your models.